F.G

NUSC Technical Report 5341

12

NUSC Technical Report 5341

ADA028497

Numerical Solutions of Initial Value Problems

Ding Lee
Systems Analysis Department

23 July 1976

NUSC
NAVAL UNDERWATER SYSTEMS CENTER
Newport, Rhode Island • New London, Connecticut

PREFACE

This work was supported by NUSC as a continuation of long-term educational support for the author's dissertation (NUSC TR 4775, May 1974). The author wishes to thank Professor Stanley Preiser, Philip Hsiang, and Robert Casal of the Polytechnic Institute of New York, Brooklyn, New York and Mr. J. L. Malone of Westinghouse Electric Corporation, Annapolis, Maryland. These men evaluated the author's technique on the IBM 360/370 and the CDC 6600 and validated his computations which had been performed on the Univac 1108.

The Technical Reviewer for this report was Dr. Sung-Hwan Ko, Code 316.

REVIEWED AND APPROVED: 23 July 1976

[illegible]

W. L. Clearwaters
Associate Technical Director
for Plans and Analysis

The author of this report is located at the New London Laboratory, Naval Underwater Systems Center, New London, Connecticut 06320.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) NUSC-TR-5341	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) NUMERICAL SOLUTIONS OF INITIAL VALUE PROBLEMS		5. TYPE OF REPORT & PERIOD COVERED (9) Technical rept.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Ding Lee		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Underwater Systems Center New London Laboratory New London, CT 06320		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE (11) 23 Jul 1976
		13. NUMBER OF PAGES (12) 118 p.
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Numerical Methods	Nonlinear Multistep Methods	Step Size
Initial Value Problems	Linear Multistep Methods	Computer Solution of Differential Equations
Differential Equations	FORTRAN Program for NLMS	
Stiff Equations		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A family of nonlinear multistep (NLMS) numerical methods has been developed that is particularly effective for solving stiff ordinary differential equations. These methods offer the advantages of avoiding small step sizes and being A-stable in the Dahlquist sense. These advantages over existing conventional and stiff methods have been demonstrated by the present author with a number of test cases. These same

→ next page

DD FORM 1 JAN 73 1473

405918 ✓ JB

20. (Cont'd)

cont. methods can also be used to solve nonasymptotically stable differential equations since they are a generalization of Linear Multistep (LMS) methods.

This report presents a detailed formulation of NLMS methods and discusses a FORTRAN V computer package specifically developed to implement NLMS methods. The package, also available in ANSI FORTRAN, is presently operational on Univac 1108, IBM 360/370, and CDC 6600 computers. Several desirable features are included in the computer program, namely, a fixed or variable step size, self-start, a selection of characteristic polynominal coefficients, predict-and-correct m times, and the inclusion of LMS methods. Whenever matrix A is a function of t, a periodic decomposition technique is employed. Otherwise, a decomposition technique can be selected by the user. Nonlinear multistep methods and their associated features constitute a powerful means for solving initial value problems of stiff, nonstiff, linear, and nonlinear ordinary differential equations. The validity of NLMS methods has been proved theoretically while the effectiveness of the computer program will be supported by the numerical evidence to be presented.

TABLE OF CONTENTS

NUMERICAL SOLUTIONS OF INITIAL VALUE PROBLEMS

1. INTRODUCTION

To solve nonstiff differential equations, the conventional Runge-Kutta and linear multistep methods are typically employed. These conventional methods are impractical for solving stiff equations because prohibitively small step sizes are required for accuracy. To overcome this difficulty, nonlinear multistep (NLMS) methods can be applied; NLMS methods have been shown to have advantages over existing stiff methods. It is desirable to be prepared with a variety of effective methods for handling both stiff and nonstiff equations. The search for such an efficient program will probably never end. To be effective, a package must be reliable, easy to modify, and convenient to use. In addition, other features are desirable. To satisfy these needs, the NLMS method was developed. It is now known that strongly stable NLMS methods are consistent and, therefore, convergent. The complete theory of NLMS methods has been published.[1]

This report will first outline some preliminary considerations and assumptions and then describe the formulation of NLMS methods along with the various built-in features. Since the principal objective was to solve stiff equations, variable-order techniques were not utilized because NLMS methods of all orders were considered to be effective.

The various features are described and illustrated by problems. A section of numerical results describes the selection of these problems from well-known sources, categorizes them in various classes, and then summarizes their characteristics in tabular form. Various NLMS methods are applied to solve these problems with given initial values. The computed solutions are next compared with exact solutions. The computational accuracy is measured mainly by relative error, which will be defined in the next section. An error of 10^{-10} is used as an upper bound to acceptable performance in test examples. Numerical results are printed out to eight significant digits. The User's Guide is intended to help the user become familiar with the inputs. To assist in this objective, the way to start a problem with various sample setups is described within each test problem. Only FORTRAN V inputs are described since ANSI FORTRAN inputs are made optional to the user. The usage of a few user-supplied or optional subroutines is also described within each test problem, which exercises various options. Programs were originally written in FORTRAN V language and checked out on the Univac 1108 computer using double-precision arithmetic by means of the EXEC 8 operating system.

These programs are also available on IBM 360/370 and CDC 6600 in FORTRAN. A version of the program is made available in ANSI FORTRAN, where the external and the adjustable dimensions were purposely eliminated. Detailed operational descriptions of the IBM and CDC machines are not included. In the appendix, a program listing of both FORTRAN V and ANSI FORTRAN are given.

2. PRELIMINARY CONSIDERATIONS

In this section, we define the problems under consideration, give the norm used for comparison, and state some assumptions.

1. Let us consider the initial value problem of a system of first-order ordinary differential equations of the form

$$\mathbf{y}' = \mathbf{f}(t,\mathbf{y}); \quad \mathbf{y}(t_o) = \mathbf{y}_o , \tag{2.1}$$

which can also be written as

$$\mathbf{y}' = \mathbf{A}\mathbf{y} + \mathbf{g}(t,\mathbf{y}) \; ; \quad \mathbf{y}(t_o) = \mathbf{y}_o \tag{2.2}$$

in the region R, defined by $-\infty < a \leq t \leq b < \infty$; $\|\mathbf{y}\| < \infty$. The matrix $\mathbf{A}$ is nonsingular, either a constant or a function of t. We assume that our initial value problems satisfy the conditions required by the existence and uniqueness theorem; therefore, $\mathbf{f}$ and $\mathbf{g}$ both satisfy a Lipschitz condition with Lipschitz constants L* and L, respectively, and $\mathbf{f}$ is continuous in R.

2. Subject to the consideration 1., we consider only first-order equations. This is not a restriction because higher order equations can always be decomposed into a system of first-order equations. Equations that are stiff, nonstiff, linear, nonlinear, homogeneous, and nonhomogeneous are all taken into consideration. We regard every problem as a system of equations.

3. Let $\mathbf{y}_c$ be a computed solution vector and $\mathbf{y}_E$ be an exact solution vector. The norm used here to measure the error is defined as follows:

$$\|\mathbf{y}\|_\infty = \lim_{p\to\infty} \|\mathbf{y}\|_p = \max_j \; |y_j| \; . \tag{2.3}$$

We define the relative error E to mean

$$E = \max_i \left| \frac{(y_i)_c - (y_i)_E}{(y_i)_E} \right| . \tag{2.4}$$

In the event that an absolute error is considered, it possesses the following definition:

$$E = \max_{i} \left\{ (y_i)_c - (y_i)_E \right\} . \qquad (2.5)$$

3. FORMULATION

The linear multistep methods of step K can be expressed by[2]

$$\sum_{i=0}^{K} \alpha_i \mathbf{y}_{n+i} = h \sum_{i=0}^{K} \beta_i \mathbf{f}_{n+i} , \tag{3.1}$$

where $\alpha_K \neq 0, |\alpha_o| + |\beta_o| > 0$, and α_i and β_i are constants independent of mesh size h. An LMS method can solve equation (2.1) effectively when $\left\| \partial \mathbf{f} / \partial \mathbf{y} \right\|$ is small.

A generalization of equation (3.1) leads fo the NLMS methods of step K in the form

$$\sum_{i=0}^{K} \alpha_i e^{\mathbf{A}h(K-i)} \mathbf{y}_{n+i} = h \sum_{i=0}^{K} \phi_{Ki}(\mathbf{A}h) \mathbf{g}_{n+i} , \tag{3.2}$$

where $\alpha_K \neq 0, |\alpha_o| + |\lambda(\phi_{Ko}(\mathbf{A}h))| > o$, and α_i are independent of h, but $\phi_{Ki}(\mathbf{A}h)$ are functions of h and nonsingular $\mathbf{A}$. (The Generalized Adams-Bashforth (GAB) and the Generalized Adams-Moulten (GAM) methods for K = 1, 2 are suggested by the work of Certaine.[3])

The NLMS methods are designed to solve equation (2) effectively when $\left\| \partial \mathbf{f} / \partial \mathbf{y} \right\| = \left\| \mathbf{A} + \partial \mathbf{g} / \partial \mathbf{y} \right\|$ is large. For $\mathbf{A}$, which is a constant matrix, we write equation (2.2) as

$$\frac{d}{dt} (e^{-\mathbf{A}t} \mathbf{y}) = e^{-\mathbf{A}t} \mathbf{g}(t, \mathbf{y}); \quad \mathbf{y}(t_o) = \mathbf{y}_o . \tag{3.3}$$

Integration of equation (3.3) over the interval $\left[t_n, t_{n+i} \right]$ gives

$$\mathbf{y}(t_{n+i}) = e^{i\mathbf{A}h} \mathbf{y}(t_n) + \int_{t_n}^{t_{n+i}} e^{\mathbf{A}(t_{n+i} - t')} \mathbf{g}(t', \mathbf{y}) \, dt' . \tag{3.4}$$

Expressing $\mathbf{g}(t', \mathbf{y})$ in a Taylor series expansion around t_n, and, next, substituting it into equation (3.4), we obtain

$$\mathbf{y}(t_{n+i}) = e^{i\mathbf{A}h}\, \mathbf{y}(t_n) + \sum_{j=0}^{\infty} \frac{\mathbf{I}_i^j(\mathbf{A}h)}{j!}\, \mathbf{g}^{(j)}(t_n, \mathbf{y}(t_n)), \tag{3.5}$$

where

$$\mathbf{I}_i^j(\mathbf{A}h) = \int_{t_n}^{t_{n+i}} e^{\mathbf{A}(t_{n+i} - t')}\, (t' - t_n)^j\, dt. \tag{3.6}$$

We define the nonlinear multistep operator $\mathcal{L}_N\,[\mathbf{y}(t); h]$ to be

$$\mathcal{L}_N\,[\mathbf{y}(t); h] = \sum_{i=0}^{K} \alpha_i\, e^{\mathbf{A}h(K-i)}\, \mathbf{y}(t + ih)$$

$$- h \sum_{i=0}^{K} \phi_{Ki}(\mathbf{A}h)\, \mathbf{g}(t + ih, \mathbf{y}). \tag{3.7}$$

Expanding $\mathbf{g}(t + ih, \mathbf{y})$ in powers of h at $t = t_n$ yields

$$\mathbf{g}(t + ih, \mathbf{y}) = \sum_{j=0}^{\infty} \frac{(ih)^j}{j!}\, \mathbf{g}^{(j)}(t_n, \mathbf{y}(t_n)). \tag{3.8}$$

If we substitute $\mathbf{g}(t + ih, \mathbf{y})$ into equation (3.7) and use equation (3.5) for $\mathbf{y}(t + ih)$ in (3.7), we obtain

$$\mathcal{L}_N[\mathbf{y}(t);h] = \sum_{i=0}^{K} \alpha_i\, e^{\mathbf{A}h(K-i)} \left[e^{i\mathbf{A}h}\mathbf{y} + \sum_{j=0}^{\infty} \frac{\mathbf{I}_i^j(\mathbf{A}h)}{j!}\mathbf{g}^{(j)} \right]$$

$$- h \sum_{i=0}^{K} \phi_{Ki}(\mathbf{A}h) \left[\sum_{j=0}^{\infty} \frac{(ih)^j}{j!}\mathbf{g}^{(j)} \right]$$

$$= \left\{ \sum_{i=0}^{K} \alpha_i\, e^{\mathbf{A}h(K-i)}\, e^{i\mathbf{A}h}\mathbf{y} \right\} + \sum_{j=0}^{\infty} \mathbf{C}_j(\mathbf{A}h)\, \mathbf{g}^{(j)},$$

where

$$\mathbf{C}_j(\mathbf{A}h) = \sum_{i=0}^{K} \alpha_i\, e^{\mathbf{A}h(K-i)} \left[\frac{\mathbf{I}_i^j(\mathbf{A}h)}{j!} \right] - h \sum_{i=0}^{K} \frac{(ih)^j}{j!}\phi_{Ki}(\mathbf{A}h).$$

A consistent NLMS method is said to be of order p if

$$\mathbf{C}_o = \mathbf{C}_1 = \dots = \mathbf{C}_p = \mathbf{0}, \text{ but } \mathbf{C}_{p+1} \neq \mathbf{0}.$$

By mathematical induction, it is seen that

$$\frac{\mathbf{A}^{j+1}\mathbf{I}_i^j(\mathbf{A}h)}{j!} = e^{i\mathbf{A}h} - \sum_{\ell=0}^{j} \frac{(i\mathbf{A}h)^\ell}{\ell!}.$$

Therefore,

$$-\mathbf{A}^{j+1}\mathbf{C}_j(\mathbf{A}h) = \sum_{i=0}^{K} \alpha_i\, e^{\mathbf{A}h(K-i)} \sum_{\ell=0}^{j} \frac{(i\mathbf{A}h)^\ell}{\ell!} +$$

$$\frac{(\mathbf{A}h)^{j+1}}{j!} \cdot \sum_{i=0}^{K} (i)^j \phi_{Ki}(\mathbf{A}h) = \mathbf{0}. \qquad (3.9)$$

A method of equation (3.2) is said to be consistent[3] if

$$\max_n \left\| \sum_{i=0}^{K} \alpha_i e^{\mathbf{A}h(K-i)} \mathbf{y}_{n+i} - h \sum_{i=0}^{K} \phi_{Ki}(\mathbf{A}h)\, \mathbf{g}_{n+i} \right\|$$

is small as $h \to 0$.

The requirement that $\mathbf{C}_j(\mathbf{A}h) = \mathbf{O}$ for $j = 0, 1, \ldots, p$ yields the consistency and permits formulating the NLMS method in the following matrix form:

$$\mathbf{E}\psi = -\mathbf{H}\mathbf{K}\phi \, , \tag{3.10}$$

where $\mathbf{E}$, ψ , $\mathbf{H}$, $\mathbf{K}$, and ϕ are described by the expanded forms of both explicit and implicit forms.

In expanded forms, the explicit schemes satisfy

$$\begin{pmatrix} I & I & \cdots & I \\ I & I+\mathbf{A}h & \cdots & I+K\mathbf{A}h \\ \vdots & \vdots & & \vdots \\ I & \sum_{m=0}^{p-1} \frac{(\mathbf{A}h)^m}{m!} & \cdots & \sum_{m=0}^{p-1} \frac{(K\mathbf{A}h)^m}{m!} \end{pmatrix} \begin{pmatrix} \alpha_0 e^{K\mathbf{A}h} \\ \alpha_1 e^{(K-1)\mathbf{A}h} \\ \vdots \\ \alpha_K I \end{pmatrix}$$

$$= - \begin{pmatrix} \frac{\mathbf{A}h}{0!} & & \bigcirc \\ & \frac{(\mathbf{A}h)^2}{1!} & \\ \bigcirc & & \ddots & \\ & & & \frac{(\mathbf{A}h)^p}{(p-1)!} \end{pmatrix} \begin{pmatrix} I & I & \cdots & I \\ \mathbf{0} & I & \cdots & (K-1)I \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & I & \cdots & (K-1)^{p-1} I \end{pmatrix} \begin{pmatrix} \phi_{K0} \\ \phi_{K1} \\ \vdots \\ \phi_{K,K-1} \end{pmatrix} \tag{3.11}$$

and the implicit schemes satisfy

$$\begin{pmatrix} I & I & \cdots & I \\ I & I + \mathbf{A}h & \cdots & I + K\mathbf{A}h \\ \vdots & \vdots & & \vdots \\ I & \sum_{m=0}^{p} \frac{(\mathbf{A}h)^m}{m!} & \cdots & \sum_{m=0}^{p} \frac{(K\mathbf{A}h)^m}{m!} \end{pmatrix} \begin{pmatrix} \alpha_0 e^{K\mathbf{A}h} \\ \alpha_1 e^{(K-1)\mathbf{A}h} \\ \vdots \\ \alpha_K I \end{pmatrix}$$

$$= - \begin{pmatrix} \frac{\mathbf{A}h}{0!} & & \bigcirc \\ & \frac{(\mathbf{A}h)^2}{1!} & \\ & & \ddots \\ \bigcirc & & & \frac{(\mathbf{A}h)^{p+1}}{p!} \end{pmatrix} \begin{pmatrix} I & I & \cdots & I \\ \mathbf{0} & I & \cdots & KI \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & I & \cdots & K^p I \end{pmatrix} \begin{pmatrix} \phi_{K0} \\ \phi_{K1} \\ \vdots \\ \phi_{KK} \end{pmatrix} . \tag{3.12}$$

We determine ϕ_{Ki} $(\mathbf{A}h)$ without loss of generality by selecting $\alpha_K = 1$ and then requiring that the condition of strong stability be realized in selecting α_i. The ϕ_{Ki} $(\mathbf{A}h)$ are determined using the above matrix formula, which can be considered as a matrix equation for the K-step, pth-order method ($K \geq 1$, $p \geq 0$). Below we show that ϕ_{Ki} $(\mathbf{A}h)$ can be determined by, first, listing the formulas for the remaining ϕ_{Ki} $(\mathbf{A}h)$, and second, exhibiting an explicit NLMS method of order 2. From formula (3.10) for nonsingular $\mathbf{H}$ and $\mathbf{K}$, we get

$$\phi = - \mathbf{K}^{-1} \mathbf{H}^{-1} \mathbf{E} \psi, \tag{3.13}$$

where ϕ is a vector of dimension (K-1) or K, depending upon whether the scheme is explicit or implicit. Explicit schemes are the following:

$$K = p = 1, \ \phi_{1,0}(\mathbf{A}h) = -(\mathbf{A}h)^{-1} (\alpha_0 e^{\mathbf{A}h} + I) \tag{3.14}$$

K = p = 2,
$$\phi_{2,0}(\mathbf{A}h) = -(\mathbf{A}h)^{-2}\left[\alpha_o(\mathbf{A}h-I)e^{2\mathbf{A}h} - \alpha_1 e^{\mathbf{A}h} - (I+\mathbf{A}h)\right] \quad (3.15)$$

$$\phi_{2,1}(\mathbf{A}h) = -(\mathbf{A}h)^{-2}\left[\alpha_o e^{2\mathbf{A}h} + \alpha_1(I+\mathbf{A}h)e^{\mathbf{A}h} + (I+2\mathbf{A}h)\right] \quad (3.16)$$

K = p = 3,
$$\phi_{3,0}(\mathbf{A}h) = -(\mathbf{A}h)^{-3}\left[\alpha_o\left(I-\frac{3}{2}\mathbf{A}h+(\mathbf{A}h)^2\right)e^{3\mathbf{A}h}\right.$$
$$\left.+\alpha_1\left(I-\frac{\mathbf{A}h}{2}\right)e^{2\mathbf{A}h} + \alpha_2\left(I+\frac{\mathbf{A}h}{2}\right)e^{\mathbf{A}h} + I + \frac{3}{2}\mathbf{A}h + (\mathbf{A}h)^2\right] \quad (3.17)$$

$$\phi_{3,1}(\mathbf{A}h) = -(\mathbf{A}h)^{-3}\left[-2\alpha_o(I-\mathbf{A}h)e^{3\mathbf{A}h} + \alpha_1(-2I+(\mathbf{A}h)^2)e^{2\mathbf{A}h}\right.$$
$$\left.-2\alpha_2(I+\mathbf{A}h)e^{\mathbf{A}h} - (2I+4\mathbf{A}h+3(\mathbf{A}h)^2)\right], \quad (3.18)$$

$$\phi_{3,2}(\mathbf{A}h) = -(\mathbf{A}h)^{-3}\left[\alpha_o\left(I-\frac{\mathbf{A}h}{2}\right)e^{3\mathbf{A}h} + \alpha_1\left(I+\frac{\mathbf{A}h}{2}\right)e^{2\mathbf{A}h}\right.$$
$$\left.+\alpha_2\left(I+\frac{3}{2}\mathbf{A}h+(\mathbf{A}h)^2\right)e^{\mathbf{A}h} + \left(I+\frac{5}{2}\mathbf{A}h+3(\mathbf{A}h)^2\right)\right]. \quad (3.19)$$

Implicit schemes are the following:

K = p = 1,

$$\phi_{1,0}(\mathbf{A}h) = -(\mathbf{A}h)^{-2}\left[\alpha_o(\mathbf{A}h-I)e^{\mathbf{A}h} - I\right], \quad (3.20)$$

$$\phi_{1,1}(\mathbf{A}h) = -(\mathbf{A}h)^{-2}\left[\alpha_o e^{\mathbf{A}h} + (I+\mathbf{A}h)\right]. \quad (3.21)$$

K = p = 2,

$$\phi_{2,0}(\mathbf{A}h) = -(\mathbf{A}h)^{-3}\left[\alpha_o\left(I-\frac{3}{2}\mathbf{A}h+(\mathbf{A}h)^2\right)e^{2\mathbf{A}h}\right.$$
$$\left.+\alpha_1\left(I-\frac{\mathbf{A}h}{2}\right)e^{\mathbf{A}h} + \left(I+\frac{\mathbf{A}h}{2}\right)\right], \quad (3.22)$$

$$\phi_{2,1}(\mathbf{A}h) = -(\mathbf{A}h)^{-3}\left[\alpha_o(-2I+2\mathbf{A}h)e^{2\mathbf{A}h}\right.$$
$$\left.+\alpha_1(-2I+(\mathbf{A}h)^2)e^{\mathbf{A}h} - 2(I+\mathbf{A}h)\right], \quad (3.23)$$

$$\phi_{2,2}(\mathbf{A}h) = -(\mathbf{A}h)^{-3}\Big[\alpha_o\left(I - \frac{\mathbf{A}h}{2}\right)e^{2\mathbf{A}h} + \alpha_1\left(I + \frac{\mathbf{A}h}{2}\right)e^{\mathbf{A}h}$$

$$+ \left(I + \frac{3}{2}\mathbf{A}h + (\mathbf{A}h)^2\right)\Big]. \qquad (3.24)$$

K = p = 3,

$$\phi_{3,0}(\mathbf{A}h) = -(\mathbf{A}h)^{-4}\Big[\alpha_o\left(-I + 2\mathbf{A}h - \frac{11}{6}(\mathbf{A}h)^2 + (\mathbf{A}h)^3\right)e^{3\mathbf{A}h}$$

$$+ \alpha_1\left(-I + \mathbf{A}h - \frac{1}{3}(\mathbf{A}h)^2\right)e^{2\mathbf{A}h} + \alpha_2\left(-I + \frac{1}{6}(\mathbf{A}h)^2\right)e^{\mathbf{A}h}$$

$$+ \left(-I - \mathbf{A}h - \frac{1}{3}(\mathbf{A}h)^2\right)\Big] \qquad (3.25)$$

$$\phi_{3,1}(\mathbf{A}h) = -(\mathbf{A}h)^{-4}\Big[\alpha_o\left(3I - 5\mathbf{A}h + 3(\mathbf{A}h)^2\right)e^{3\mathbf{A}h}$$

$$+ \alpha_1\left(3I - 2\mathbf{A}h - \frac{1}{2}(\mathbf{A}h)^2 + (\mathbf{A}h)^3\right)e^{2\mathbf{A}h}$$

$$+ \alpha_2\left(3I + \mathbf{A}h - (\mathbf{A}h)^2\right)e^{\mathbf{A}h}$$

$$+ \left(3I + 4\mathbf{A}h + \frac{3}{2}(\mathbf{A}h)^2\right)\Big] \qquad (3.26)$$

$$\phi_{3,2}(\mathbf{A}h) = -(\mathbf{A}h)^{-4}\Big[\alpha_o\left(-3I + 4\mathbf{A}h - \frac{3}{2}(\mathbf{A}h)^2\right)e^{3\mathbf{A}h}$$

$$+ \alpha_1\left(-3I + \mathbf{A}h + (\mathbf{A}h)^2\right)e^{2\mathbf{A}h}$$

$$+ \alpha_2\left(-3I - 2\mathbf{A}h + \frac{1}{2}(\mathbf{A}h)^2 + (\mathbf{A}h)^3\right)e^{\mathbf{A}h}$$

$$+ \left(-3I - 5\mathbf{A}h - 3(\mathbf{A}h)^2\right)\Big] \qquad (3.27)$$

$$\phi_{3,3}(\mathbf{A}h) = -(\mathbf{A}h)^{-4}\Big[\alpha_o\left(I - \mathbf{A}h + \frac{1}{3}(\mathbf{A}h)^2\right)e^{3\mathbf{A}h} + \alpha_1\left(I - \frac{1}{6}(\mathbf{A}h)^2\right)e^{2\mathbf{A}h}$$

$$+ \alpha_2\left(I + \mathbf{A}h + \frac{1}{3}(\mathbf{A}h)^2\right)e^{\mathbf{A}h} + \left(I + 2\mathbf{A}h + \frac{11}{6}(\mathbf{A}h)^2 + (\mathbf{A}h)^3\right)\Big] \qquad (3.28)$$

In the explicit case when K = p = 2, it is seen that

$$\mathbf{H} = \begin{pmatrix} \mathbf{A}h & \mathbf{O} \\ \mathbf{O} & (\mathbf{A}h)^2 \end{pmatrix}, \mathbf{H}^{-1} = (\mathbf{A}h)^{-2} \begin{pmatrix} \mathbf{A}h & \mathbf{O} \\ \mathbf{O} & I \end{pmatrix};$$

$$\mathbf{K} = \begin{pmatrix} I & I \\ \mathbf{O} & I \end{pmatrix}, \mathbf{K}^{-1} = \begin{pmatrix} I & -I \\ \mathbf{O} & I \end{pmatrix};$$

$$\mathbf{E} = \begin{pmatrix} I & I & I \\ I & I + \mathbf{A}h & I + 2\mathbf{A}h \end{pmatrix};$$

and, therefore,

$$\phi = \begin{pmatrix} \phi_{2,0} \\ \phi_{2,1} \end{pmatrix} = -(\mathbf{A}h)^{-2} \begin{pmatrix} \alpha_0 (\mathbf{A}h - I)e^{2\mathbf{A}h} - \alpha_1 e^{\mathbf{A}h} - \alpha_2 (I + \mathbf{A}h) \\ \alpha_0 e^{2\mathbf{A}h} + \alpha_1 (I + \mathbf{A}h)e^{\mathbf{A}h} + \alpha_2 (I + 2\mathbf{A}h) \end{pmatrix} \tag{3.29}$$

The selection of $\alpha_2 = 1$, $\alpha_1 = -1$, and $\alpha_0 = 0$ leads to the GAB method which gives

$$\phi = \begin{pmatrix} \phi_{2,0} \\ \phi_{2,1} \end{pmatrix} = -(\mathbf{A}h)^{-2} \begin{pmatrix} e^{\mathbf{A}h} - (I + \mathbf{A}h) \\ -(I + \mathbf{A}h)e^{\mathbf{A}h} + (I + 2\mathbf{A}h) \end{pmatrix}. \tag{3.30}$$

However, the eigenvalues in stiff equations differ greatly in magnitude. As a consequence, double-precision arithmetic should be used in calculating ϕ.

4. STEP SIZES OF NLMS METHODS

Nonlinear multistep methods are designed to avoid the use of small stepsizes where $\mathbf{g}(t,\mathbf{y})$ is a slowly varying function that can be approximated by a low-order polynomial in t. To demonstrate this, we give an analysis below and show that a larger step size can be selected using NLMS methods than using LMS methods. From equation (3.2), since $\alpha_k \neq 0$, we can write

$$\mathbf{y}_{n+k} = \frac{1}{\alpha_K}\left\{ -\sum_{i=0}^{K-1} \alpha_i e^{\mathbf{A}h(K-i)} \mathbf{y}_{n+i} + h \sum_{i=0}^{K} \phi_{Ki}(\mathbf{A}h)\mathbf{g}_{n+i} \right\}, \tag{4.1}$$

which is of the form

$$\mathbf{y} = G(\mathbf{y}),$$

where $\mathbf{y} = \mathbf{y}_{n+k}$. The successive iterative form gives

$$\mathbf{y}^{(\gamma+1)} = G(\mathbf{y}^{(\gamma)}) \tag{4.2}$$

for any initial vector $\mathbf{y}^{(0)}$.

Let $\mathbf{G}(\mathbf{y})$ be defined for $\|\mathbf{y}\| < \infty$, and let there exist a constant k such that $0 \leq k < 1$. Then, $\mathbf{G}(\mathbf{y})$ satisfies the condition

$$\|G(\mathbf{y}^*) - G(\mathbf{y})\| < k\|\mathbf{y}^* - \mathbf{y}\| . \tag{4.3}$$

Using the definition of $G(\mathbf{y})$, formula (4.1), and the fact that $\mathbf{g}(t,\mathbf{y})$ satisfies the Lipschitz condition with Lipschitz constant L, we see that condition (4.3) is satisfied by

$$k = \frac{h\|\phi_{KK}(\mathbf{A}h)\|}{\alpha_K} L \tag{4.4}$$

for sufficiently small h and for all $\|\mathbf{A}\| < \infty$.

For the iterative procedure (4.2) to converge for arbitrary initial $\mathbf{y}^{(0)}$, k is required to be less than 1:

$$k < 1 \longrightarrow \frac{h \left\| \phi_{KK} (\mathbf{A}h) \right\|}{\alpha_K} L < 1. \tag{4.5}$$

Conventionally, when using LMS K-step methods with $\alpha_K = 1$, we select h such that

$$\beta_K \, h \, L^* \sim k \;\; (< 1). \tag{4.6}$$

Similary, for NLMS K-step methods, we select h_N to satisfy condition (4.5):

$$\left\| \phi_{KK} (\mathbf{A}h_N) \right\| \;\; h_N \, L \sim k . \tag{4.7}$$

Combining (4.6) and (4.7), we see that

$$h_N = \left\| \phi_{KK}{}^{-1} (\mathbf{A}h_N) \right\| \; \beta_K \; \frac{L^*}{L} \; h. \tag{4.8}$$

For $\left\| \phi_{KK}{}^{-1} (\mathbf{A}h_N) \right\| \; \beta_K$ not too small, we know that $L^* >> L$; therefore, $h_N >> h$. This tells us that we can choose a much larger step size using NLMS then we can choose using LMS.

The quantity h_N can be selected to satisfy

$$h_N < \frac{1}{\left\| \phi_{KK} (\mathbf{A}h_N) \right\| \cdot \left\| \frac{\partial \mathbf{g}}{\partial \mathbf{y}} \right\|} . \tag{4.9}$$

In the event that $\left\| \frac{\partial \mathbf{g}}{\partial \mathbf{y}} \right\| = 0$, we can use a large h_N. This advantage is demonstrated by the problem below.

We use problem 1 from section 8.3 to demonstrate the step size choice. The problem is

$$\mathbf{y}' = -100\mathbf{y} + (1 + t^2); \; \mathbf{y}(0) = 1 .$$

Table 4-1 compares two methods of solution.

Table 4-1. Comparison of Two Methods for Solving Problem 1

	Adams-Bashforth	Explicit NLMS	Remarks
$\left\| \frac{\partial \mathbf{F}}{\partial \mathbf{y}} \right\|$	100	0	$\mathbf{F} = \begin{cases} \mathbf{f}(t, \mathbf{y}) & \text{AB} \\ & \text{for} \\ \mathbf{g}(t, \mathbf{y}) & \text{NLMS} \end{cases}$
Step Number	3	3	
Step Size	2^{-8}	2.5	$h_N = 640h$
Tmax	10	10	
Solution Vector	.1008 0020 + 01	.1008 0020 + 01	
Exact Solution	.1008 0020 + 01		
Total Steps	2560	1	

5. FEATURES

5.1 VARIABLE STEP SIZE AND FIXED STEP SIZE

Variable step size is a desirable feature that is incorporated into this approach. By controlling the step size, the solution may be obtained quicker through the use of a larger step size. In some instances, the user may desire to examine the solution at a specified time value. The variable-step-size technique may not meet this requirement because a larger step can often bypass that point. This is commonly encountered in all variable-step-size programs. Although NLMS methods are designed to avoid the use of small step sizes, the program is designed with this option. However, the user can still use the variable step size if he desires and select the maximum step size HMAX.

To exercise this option, we require that the indicator IPC be defined as follows:

$$\text{IPC} = 0\text{: fixed step size with INDEX} = \begin{cases} 0 \text{ explicit} \\ 1 \text{ implicit} \end{cases}$$

$$\text{IPC} \neq 0\text{: variable step size; PC}^m.$$

If the present step size h needs a change, it is changed by an amount rh where $1/2 \leq r \leq 2$. To save computing time, we change h by halving or doubling, depending upon the need. This is handled automatically by the program.

An example to demonstrate this feature is given in problem 1 of section 8.3. Closely connected with the variable-step-size procedure is the PC^m procedure, which will be described in the next section.

5.2 THE PC^m PROCEDURE

The PC^m procedure stands for "predict and correct m times." This procedure has been widely used to achieve efficiency. Henrici[2] discussed a single correction. Others feel that correction may be needed more than once. It is frequently true that a good predictor needs but a single correction. In this package, we set the upper bound of m to be 3.

It should be noted that when NLMS methods are applied to solve stiff equations, when **g** is a function of t alone or is a constant, the PC^m procedure may not be needed.

5.3 SELF-STARTING

The possibility of missing starting values is common with all multistep methods. The missing values have to be provided by an independent method. The Runge-Kutta and Adam-Bashforth methods are often used as starters. The self-starting procedure is provided here by using NLMS or LMS first-order methods. As mentioned previously, NLMS methods allow the use of large step size to solve stiff equations; therefore, we shall adopt the explicit NLMS methods of order one as a starter for solving stiff equations. The Adams-Bashforth first-order method is used as a starter for solving nonstiff equations, which requires the use of a very small step size. In the above case, LMS methods are called for and the self-starting procedure requested. It is suggested that the user specify a small initial step size to achieve satisfactory accuracy.

In most instances, the use of NLMS explicit methods of order one to self-start is recommended because these methods are a good starter and, most of all, because the step size is not restricted; the only requirement is that the **g** function must be defined. To do this, a simple modification can be incorporated in the START subroutine. The user can save the indicator, METHOD, from the argument and set it to 1 after entry, then reset METHOD to the original state before RETURN.

5.4 SELECTION OF CHARACTERISTIC COEFFICIENTS α_i

The characteristic polynomial of LMS methods takes the form

$$\rho(\zeta) = \sum_{i=0}^{K} \alpha_i \zeta^i = 0, \; \alpha_K = 1. \tag{5.1}$$

Let ζ_j be the roots of (5.1). The relationship

$$\prod_{j=1}^{K} (\zeta - \zeta_j) = \sum_{i=0}^{K} \alpha_i \zeta^i \tag{5.2}$$

enables us to express α_i as a function of ζ_j; the ζ_j determine the root condition and, thus, the stability. In this manner, after the ζ_j are chosen to satisfy the root condition of stability, the α_i can be determined.

For NLMS methods, first we determine $\phi_{Ki}(\mathbf{A}h)$, which are dependent on α_i, as given by formulas (3.14) - (3.28). After $\phi_{Ki}(\mathbf{A}h)$ are determined, the numerical solution is calculated by means of formula (3.2), which is dependent on α_i.

For LMS methods, the α_i are usually tabulated. One should note that, in formula (2.2), if $\mathbf{A} = \mathbf{O}$, NLMS methods reduce to LMS methods. If we multiply (3.9) by $\left[(\mathbf{A}h)^{j+1}\right]^{-1}$ and let $\|\mathbf{A}\| \rightarrow 0$, we get

$$\sum_{i=0}^{K} \alpha_i \frac{i^{j+1}}{(j+1)!} = \frac{1}{j!} \sum_{i=0}^{K} i^j \phi_{Ki}(\mathbf{O}) . \tag{5.3}$$

For j=0, 1,...,p-1, equation (5.3) gives a formulation of the LMS methods of order p. The matrix form is

$$\begin{pmatrix} 0 & 1 & \cdots & K \\ 0 & \frac{1^2}{2!} & \cdots & \frac{K^2}{2!} \\ \vdots & \vdots & & \vdots \\ 0 & \frac{1^p}{p!} & \cdots & \frac{K^p}{p!} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_K \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \cdots & K \\ \vdots & \vdots & & \vdots \\ 0 & \frac{1^{p-1}}{(p-1)!} & \cdots & \frac{K^{p-1}}{(p-1)!} \end{pmatrix} \begin{pmatrix} \phi_{K0} \\ \phi_{K1} \\ \vdots \\ \phi_{KK} \end{pmatrix} . \tag{5.4}$$

Using (5.4), we can express $\beta_i \; (= \phi_{Ki}(\mathbf{O}))$ as a function of α_i. We proceed by using the explicit LMS methods of order 2 whose β coefficients are

$$\begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_1/2 \\ \alpha_1/2 + 2\alpha_2 \end{pmatrix} . \tag{5.5}$$

The selection of two roots, 0 and 1, leads to strong stability and gives $\alpha_0 = 0$, $\alpha_1 = -1$ and $\alpha_2 = 1$. This implies that

$$\begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} -1/2 \\ 3/2 \end{pmatrix},$$

which gives the Adams-Bashforth method of order two. The author is unaware of any program that allows a user to select α_i. One does not need to make a choice when using tabulated α_i values. He has found[1] that the selection of roots of the characteristic polynomial can minimize the initial local discretization error C_{p+1}. Therefore, the selection of α_i may aid in studying the minimization of C_{p+1}. This is still an open problem. Hence, this option may not be immediately exercised, but it is available when needed.

Different characteristic roots satisfying the root condition give the characteristic coefficients α_i. Different families of implicit NLMS methods of order two are selected to solve problem 3 in section 8. These families, their characteristic roots, and the associated characteristic coefficients are identified by the following:

1. Strongly Stable Generalized-Adams Family

   characteristic roots: 0, 1
   characteristic coefficients: $\alpha_0 = 0$, $\alpha_1 = -1$, $\alpha_2 = 1$.

2. Strongly Stable Family

   characteristic roots: 1/2, 1
   characteristic coefficients: $\alpha_0 = 1/2$, $\alpha_1 = -3/2$, $\alpha_2 = 1$

3. Weakly Stable Generalized-Milne Family

   characteristic roots: -1, 1
   characteristic coefficients: $\alpha_0 = -1$, $\alpha_1 = 0$, $\alpha_2 = 1$.

5.5 $\mathbf{A}$ AS A FUNCTION OF t

Consider the initial value problem

$$\mathbf{y}' = \mathbf{A}(t)\,\mathbf{y} + \mathbf{g}(t, \mathbf{y});\quad \mathbf{y}(t_0) = \mathbf{y}_0 \tag{5.6}$$

over the interval $\left[t_0, t_0 + nh\right]$. We periodically decompose equation (5.6) into

$$\mathbf{y}' = \mathbf{A}(t_i)\,\mathbf{y} + \bar{\mathbf{g}}(t, \mathbf{y});\quad \mathbf{y}(t_0) = \mathbf{y}_0, \tag{5.7}$$

where

$$\bar{\mathbf{g}}(t, \mathbf{y}) = \left\{\mathbf{A}(t) - \mathbf{A}(t_i)\right\}\mathbf{y} + \mathbf{g}(t, \mathbf{y}), \tag{5.8}$$

such that Re $\left\{\lambda\,(\mathbf{A}(t_i))\right\} < 0$. We periodically evaluate $\mathbf{A}(t)$ at $t = t_i$. The requirement that Re $\{\lambda\,(\mathbf{A}(t_i))\} < 0$ can be realized by construction and can be assured since Re $\{\lambda\,(\mathbf{A}(t))\} < 0$ for all t. The property of $\bar{\mathbf{g}}(t, \mathbf{y})$ can still be maintained for small h since $\|\mathbf{A}(t) - \mathbf{A}(t_i)\|$ is small and can be controlled by the step size. In cases where the user possesses prior knowledge about the problem, he can introduce a constant $\mathbf{A}$ and use the same decomposition technique and perhaps obtain quicker and more precise results. This is not to say decomposition is unique.

5.6 INCLUSION OF LMS METHODS

As a part of this package LMS methods are included, whose step numbers are chosen to be compatible with the same step numbers used for NLMS methods that do not exceed order three. The application of LMS methods is indicated by an indicator, METHOD, which has the following definition:

$$\text{METHOD} = \begin{cases} 0, & \text{LMS methods} \\ 1, & \text{NLMS methods.} \end{cases}$$

The inclusion of LMS methods is merely an option. The reader is asked to consult reference 4 for an efficient implementation of the Adams methods. However, the LMS methods incorporated in this package cover a complete linear multistep family rather than an Adams family alone.

5.7 MISCELLANEOUS FEATURES

1. For solving stiff equations, if $\mathbf{g}(t, \mathbf{y}) = \mathbf{g}(t)$. i.e., independent of $\mathbf{y}$, no correction is needed. To exercise this feature. set the indicator IPC equal to zero so that unnecessary computations are avoided.

2. If the stiff equation to be solved is homogeneous and $\mathbf{A}$ is a constant matrix, there is no need to calculate $h \sum_{i=0}^{K} \phi_{Ki}(\mathbf{A}h)\mathbf{g}_{n+i}$. By setting the indicator IGFN equal to zero, the program will bypass the $h \sum_{i=0}^{K} \phi_{Ki}(\mathbf{A}h)\mathbf{g}_{n+i}$ computation loop. In the event that $\mathbf{A}$ is a function of t, even if the system is homogeneous, then $\mathbf{g}(t, \mathbf{y}) = \mathbf{0}$ must be defined in the GFN subroutine.

3. A user's starter can be supplied to replace the START through either the argument or the complete replacement of the subroutine.

Subroutine INVERT can be treated in the same manner as above.

6. PROGRAMMING ASPECTS

6.1 NOTES TO USERS

1. Double precision arithmetic is used for all computations.

2. Subroutines not used by the user can be ignored. They are well-defined internally.

3. Subroutines that are required to be supplied by the user must not be ignored in order to obtain correct computations.

   a. If NLMS methods are used, GFN must be supplied, except $\mathbf{g}(t,y) = \mathbf{0}$(IGFN = 0).

   b. If LMS methods are used, FFN must be supplied.

   c. If $\mathbf{A}$ is a function of t, AFNT must be defined.

4. The solution vector is either in YNEW(I) or Y(KSTEP + 1, I). The associated t_i is in TEA. The user who desires to output his results can incorporate the desired output format into PRINT subroutine.

5. The existing Univac 1108 EXEC 8 library matrix inversion subroutine DGJR is used for test problems.

6. Certain detections to inputs are provided in FORTRAN V programs. Default values are assigned to protect a great number of inputs. This advantage can be used since the inputs are defined on NAMELIST. However, this advantage is not applicable to ANSI FORTRAN users; their inputs must be supplied correctly.

7. Meanings of a few special indicators in DIFEQ:

   Indicators, brought to the DIFEQ program through the CALL argument, are IG, IV, INDEX, IAT. These are normal indicators.

$$\left.\begin{array}{l}\text{IG}\\ \text{IV}\\ \text{INDEX}\\ \text{IAT}\end{array}\right\} \begin{array}{c}\text{indicates whether or}\\ \text{not}\end{array} \left\{\begin{array}{l}\mathbf{g}(t,\mathbf{y}) = \mathbf{0}\ .\\ \text{a variable step size is used.}\\ \text{a method is explicit.}\\ \mathbf{A}\text{ is a function of t.}\end{array}\right.$$

Other than these indicators, there are a few indicators specially used by DIFEQ to produce computational efficiency and accuracy. Their meanings need to be explained.

LMT. Normal use is to define LMT = m = 3 when a variable-step-size, PC^m procedure is used. To ensure this, LMT is set to 3 initially. In the event a fixed-step-size, implicit method is asked for, we make use of this indicator to yield the following control.

a. When IV = 0 and INDEX = 1, a fixed-step-size, implicit method is asked for.

b. After a complete calculation of the required implicit formula, LMT is changed to contain 1. The purpose of this modification is to inform the program to predict the next $\mathbf{y}_n$ value in order to implicitly calculate the $\mathbf{y}_n$. This internal modification does not affect the subsequent calculations.

ITER. A counter for every successful corrector's convergence or every fixed-step calculation. To take advantage of this, this indicator is designed to appear in the argument of PRINT, so that the user may use this information to control his printouts. As an example, if the user wants to print out results at every 100 successful calculations, he needs to define a statement in the PRINT program that reads:

```
IF(MOD(ITER,100) .EQ. 0)WRITE(4,12)H,T,(Y(L),L=1,N).
```

ISTEP. If a fixed step size is used to proceed for subsequent computations, the matrix exponentials and the determination of ϕ_{Ki} $(\mathbf{A}h)$ can be calculated only once. When a variable step size is used, it is necessary to calculate the matrix exponentials and to determine the ϕ_{Ki} $(\mathbf{A}h)$ for every h. When the program finds a successful h that leads to the corrector's convergence as well as satisfying user's tolerance, we take a new approach: We can use the same h to proceed, if required conditions are met, for at most three calculations; we then consider doubling the step size. Suppose we double the step size too soon. Then it is necessary to calculate the matrix exponentials and to determine the $\phi_{Ki}(\mathbf{A}h)$. If this new h does not lead to the corrector's convergence, all above computations are wasteful. Then we need somehow to use the matrix exponentials and the ϕ_{Ki} $(\mathbf{A}h)$ for the old h. Without worrying about memory space, the old values can be saved; without worrying about computation time, the old values can be recalculated. Certainly, we cannot save both the time and the space simultaneously. Now, the reason that ISTEP is used to minimize cost becomes apparent. In the case of solving stiff equations, chances are good for success, even if h is doubled. The user can take advantage of this by requiring the limit of ISTEP to be 1. The change to the program is almost trivial. The present setup doubles h after each successful calculation.

IMIN. To perform floating-point addition between T(1) and H, the difference between T(1) and H must be within the allowable significant digits of the computer. To maintain m significant digits in floating-point addition, the exponent (T(1)) minus the exponent (H) cannot exceed m. We define HMIN to mean h_{min} and to satisfy this range. In reality, even if the Lipschitz constant is large and even if the LMS methods are called for, HMIN will not fall outside of the significant digit range, and we can assign a larger h_{min} to avoid excessive computations. Once this safeguard is built in, when H reaches HMIN more than three times, we will not continue. We will terminate the program with a message for the user to tell him what to do. The variable IMIN is defined to count how many times H reached HMIN.

8. Even though programs work for the same computer in different locations, the input/output (I/O) units may differ in number; users are advised to make sure that the unit numbers are in agreement with the program.

6.2 FORMATS OF USER-SUPPLIED SUBROUTINES

A few optional subroutines must be supplied by the user under certain conditions. These conditions are listed below along with their required formats.

1. INVERT. This is a matrix inversion subroutine, called by NLMS methods whose CALL statement has the format

CALL INVERT (A, N, B),

where

A is an (n x n) matrix

N is the order of matrix A

B contains the A inverse after exit.

As an example, using an existing Univac 1108 library subroutine DGJR (Gauss-Jordan Reduction), the structure of INVERT should look like

```
   SUBROUTINE INVERT (A, N, ANS)
   PARAMETER NR=20, NC=20
   IMPLICIT REAL*8 (A-H, O-Z)
   DIMENSION A (NR, NC), ANS (NR, NC)
   DIMENSION V (2), JC (NC)
   V (1) = 1.D0
   D0 1 I = 1, N
   D0 1 J = 1, N
 1 ANS (I, J) = A (I, J)
   CALL DGJR (ANS, NR, NC, N, N, $10, JC, V)
   RETURN
10 WRITE (4, 2)
 2 FORMAT (3X, 'MATRIX INVERSION ERROR')
   RETURN
   END
```

2. <u>BEGIN</u>. This subroutine supplies initial values. It is required if the self-starting procedure is not used. Its CALL statement has the format

```
FORTRAN V:  CALL  BEGIN (K, H, N, Y, YN, YOLD, INVERT,
             IT, METHOD, IAT)
ANSI FORTRAN:  CALL START (K, H, N, Y, YN, YOLD, IT,
                METHOD, IAT),
```

where

K is the step number

H is the step size

N is the order of the system

Y is a two-dimensional array containing the initial values; at most, 4 rows, 20 columns.

YOLD is the same dimension as Y, used as a working storage.

INVERT is the name of a matrix inversion subroutine.

IT is an indicator that indicates whether the calculation of a **g**-function is needed.

METHOD indicates use of either NLMS or LMS explicit of order one.

IAT indicates whether **A** is a function of t.

As an example, if one uses exact initial values for problem 5, section 8, the structure of BEGIN should look like this in FORTRAN V:

```
SUBROUTINE BEGIN (K,H,N,Y,YN,YOLD,INVERT,IT,METHOD,IAT)
PARAMETER NR = 20, NC = 20
IMPLICIT REAL*8 (A=H, O-Z)
EXTERNAL INVERT
COMMON A (NR, NC), ALPHA (4), T (NC)
DIMENSION Y (4, NC), YN (N), YOLD (4, NC)
H2 = T (1) + H
H3 = H2 + H
Y(2,1) = DEXP (-H2*H2/2.) - DEXP (-H2) + 1.0
Y(3,1) = DEXP (-H3*H3/2.) - DEXP (-H3) + 1.0
RETURN
END.
```

In ANSI FORTRAN, the structure should be as follows:

```
SUBROUTINE START (K,H,N,Y,YN,YOLD,IT,METHOD,IAT)
COMMON A (20, 20), ALPHA (4), T (20)
DIMENSION Y (4,20), YN (20), YOLD (4,20)
DOUBLE PRECISION A, ALPHA, H, T, Y, YN, YOLD, H2, H3
H2 = T (1) +H
H3 = H2 + H
Y (2, 1) = DEXP (-H2*H2/2.D0) -DEXP (-H2) +1.D0
Y (3,1) = DEXP (-H3*H3/2.D0) -DEXP (-H3) +1.D0
RETURN
END
```

3. <u>GFN</u>. This subroutine calculates the **g**(t, **y**) at t = t_i for every i. It is required by NLMS methods only if the differential equation is non-homogeneous. Its CALL statement has the following format:

```
FORTRAN V:  CALL GFN (G, H, N, Y, K, T, A, NR, NC)
ANSI FORTRAN:  CALL GFN (G, H, N, Y, K, T, A),
```

where

G is a vector that contains calculated **g** function values

H equals the step size

N is the order of the system

Y is a two-dimensional array Y (I, J):$1 \leq I \leq 4$;; $1 \leq J \leq N$

K equals the step number

T is a vector that contains time values

A is an (n x n) matrix

NR equals the number of rows of A

NC is the number of columns of A.

For example, to solve a system of two stiff equations, problem 3 of section 8, the structure of the GFN should be as follows in FORTRAN V:

```
SUBROUTINE GFN (G,H,N,Y,K,T,A,NR,NC)
IMPLICIT REAL*8 (A-H, O-Z)
DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)
G (1) = 1.0
G (2) = 1.0
RETURN
END
```

The structure in ANSI FORTRAN should be as follows:

```
SUBROUTINE GFN (G,H,N,Y,K,T,A)
DOUBLE PRECISION A (20, 20), Y (4, 20), G (20), T (20), H
G (1) = 1.D0
G (2) = 1.D0
RETURN
END
```

4. FFN. This FFN subroutine calculates $\mathbf{f}(t, \mathbf{y})$, which is required only by LMS methods. Its CALL statement has the format

```
CALL FFN (Y,N,FN,I),
```

where

Y is a 2-dimensional array Y (I, J):

$$1 \leq I \leq \text{step number} + 1\ ;\ 1 \leq J \leq N$$

N equals the order of the system

FN is a vector to store calculated $\mathbf{f}$ function values

I is the intermediate step index .

For example, to solve a system of two equations, problem 4 of section 8, using LMS methods, the structure of FFN should be as follows for FORTRAN V:

```
SUBROUTINE FFN (Y,N,FN,I)
PARAMETER NR = 20, NC =20
IMPLICIT REAL*8 (A-H, O-Z)
COMMON A (NR, NC), ALPHA (4), T (NC)
DIMENSION FN (1), Y (4, NC)
FN (1) = Y (I, 2)
FN (2) = -Y (I, 1)
RETURN
END
```

For ANSI FORTRAN, the structure is

```
SUBROUTINE FFN (Y,N,FN,I)
COMMON A (20, 20), ALPHA (4), T (20)
DIMENSION FN (20), Y (4, 20)
DOUBLE PRECISION A, ALPHA, FN, T, Y
FN (1) = Y (I, 2)
FN (2) = -Y (I, 1)
RETURN
END
```

5. AFNT. This AFNT subroutine should be supplied by the user when $\mathbf{A}$ is a function of t. This subroutine calculates every $\mathbf{A}$ (t) at $t = t_i$. Its CALL statement has the format

FORTRAN V: CALL AFNT (A,N,T,NR,NC)
ANSI FORTRAN: CALL AFNT (A,N,T),

where

A is a two-dimensional array, A (NR, NC)

N equals the order of the matrix

T is a time variable

NR is the number of maximum rows of $\mathbf{A}$

NC equals the number of maximum columns of $\mathbf{A}$.

For example, to solve a system of one equation (problem 5) using an NLMS method of order 2, the structure of AFNT in FORTRAN V should be:

```
SUBROUTINE AFNT (A, N, T, NR,NC)
DOUBLE PRECISION A (NR, NC), T
A (1, 1) = -T
RETURN
END
```

In ANSI FORTRAN, it should be:

```
SUBROUTINE AFNT (A, N, T)
DOUBLE PRECISION A (20, 20), T
A (1,1) = -T
RETURN
END
```

6. PRINT. This PRINT is designed for the user's convenience to output the results in his desired formats. Its CALL statement has the format

CALL PRINT (H,T,Y,N,I),

where

H is the present step size

T is the present time step

Y is a one-dimensional array containing the current solution

N is the order of the system

I is an iteration counter.

For example, results are required to be printed out at t = 5 and 10 of problem 5 (section 8) along with exact solutions at the same time values. The PRINT statement can be designed as follows:

```
       SUBROUTINE PRINT (H,T,Y,N,I)
       DOUBLE PRECISION H, T, Y (1)
    12 FORMAT (1X, 2 (E15.8, 2X), 4X, 5 (E15.8, 2X))
       IF (T-4. 995D0) 84,84,83
    83 IF (T-5. 005D0) 1184,1184,85
    85 IF (T-9. 995D0) 84,84,86
    86 IF (T-10. 005D0) 1184,1184,84
  1184 WRITE (4,12) H,T, (Y(J), J = 1, N)
       CALL EXACT (T,Y)
    84 RETURN
       END
```

7. USER'S GUIDE

The following table gives the options, the default conditions, and the input indicators available.

Table 7-1. User's Guide

Option	Default Condition	Input Indicator
$\mathbf{A}$ is a function of t	0	IAT $\neq$ 0
ERR	0.1D-11	
F-function (FFN)	0	Called by LMS only if METHOD = 0
final time		TMAX
fixed step size		IPC = 1 and INDEX = $\begin{cases} 0 \text{ Explicit} \\ 1 \text{ Implict} \end{cases}$
G-function (GFN)	0	IGFN $\begin{cases} = 0 \text{ not needed} \\ \neq 0 \text{ need for stiff eqs.} \end{cases}$
INDEX	0	= $\begin{cases} 0 \text{ Explict} \\ 1 \text{ Implitict} \end{cases}$
initial time		T (1)
INVERT		user-supplied matrix inversion

Table 7-1. User's Guide (cont)

Option	Default Condition	Input Indicator
KSTEP, step number	1	KSTEP
METHOD	1	$= \begin{cases} 0 \text{ LMS} \\ 1 \text{ NLMS} \end{cases}$
PC^m	0	IPC = 1 or IAT $\neq$ 0, m = 3
START	self-starting	User-supplied starter
step size	0.01	H
step size (maximum)	0.1	HMAX
variable step size		IPC = 1
initial vector		YZERO
α_i	GAB-GAM	ALPHA
order of the system		N

8. NUMERICAL RESULTS AND INPUT SETUPS

8.1 PROBLEM IDENTIFICATION

An ordinary differential equation is said to belong to the class (N, S, H, L) if it satisfies the following definitions:

$$N = \begin{cases} 0 \text{ first order} \\ 1 \text{ higher order} \end{cases}$$

$$S = \begin{cases} 0 \text{ nonstiff} \\ 1 \text{ stiff} \end{cases}$$

$$H = \begin{cases} 0 \text{ homogeneous} \\ 1 \text{ nonhomogeneous} \end{cases}$$

$$L = \begin{cases} 0 \text{ linear in} \\ 1 \text{ nonlinear in.} \end{cases}$$

For example, the equation

$$\mathbf{y}' = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{y} \, ; \quad \mathbf{y}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

is a first-order, nonstiff, homogeneous, linear differential equation. Therefore it belongs to (0, 0, 0, 0).

8.2 SELECTION OF TEST PROBLEMS

Seven test problems have been selected from various sources[6-11] in order to illustrate the different options. At least one feature is exercised in each problem. The exactstart will be used for most test problems. The formats of user-supplied subroutines are described within each problem in FORTRAN V language. The main body of ANSI FORTRAN is the same as FORTRAN V of each subroutine; therefore, the ANSI FORTRAN version is omitted. For exact ANSI FORTRAN formats, the user can consult the ANSI FORTRAN in the appendix. Table 8-1 itemizes the seven problems, which are then each discussed in detail.

8.3 PROBLEMS, INPUT SETUPS, AND SOLUTIONS

Problem 1

$$\mathbf{y}' = -100\ \mathbf{y} + (1 + t^2);\ \mathbf{y}(0) = 1.$$

Exact Solution

$$\mathbf{y}(t) = \left(1 - \frac{1}{100} - \frac{2}{100^3}\right) e^{-100t} + \frac{1}{100} + \frac{1}{100^3} (100^2\, t^2$$

$$-200t + 2).$$

Classification

$$(0,1,1,0).$$

Eigenvalues of **A**

$$\{-100\}$$

Method Applied

Implicit NLMS of order 3 with fixed step size. Linear multistep of order 3 with PC^m procedure.

Step Size Used

1.25 and 2.5 for NLMS. Variable step size for LMS with initial h = 0.01.

Time Interval

$$[0,\ 10]$$

Special Features Exercised

Fixed step size.
Variable step size.

Table 8-1. The Sevcn Test Problems

Problem No.	Order of System	Class	Eigenvalues	$\mathbf{A}(t)$	$\mathbf{g}(t,\mathbf{y})$	Options Applied	Source
1	1	(0,1,1,0)	real		$\mathbf{g}(t)$	Implicit NLMS and LMS of order 3	Lee[1]
2	2	(0,1,1,0)	real		$\mathbf{0}$	Implicit NLMS of order 2	Weaver[10]
3	2	(0,0,1,0)	real		constant	Explict NLMS of order 3	Lee[1]
4	2	(1,0,0,0)	purely imaginary		$\mathbf{0}$	Explict LMS of order 2	Krough[9]
5	1	(0,0,1,0) (0,1,1,0)	real	X	$\mathbf{g}(t)$	NLMS of order 3	Krough[9]
6	1	(1,1,1,1)	real		$\mathbf{g}(t,\mathbf{y})$	NLMS of order 3	Jain[8]
7	4	(0,1,1,0)	real		$\mathbf{g}(t)$	NLMS or order 1	Gear,[7] Krough[9]

Special Features Exercised (cont)

Exact start.
PC^m procedure.

FORTRAN V Inputs Set-Up

Fixed step size, implicit NLMS of order 3.

```
 $INPUTS
  N=1, KSTEP=3, ALPHA(1)=0.D0, 0.D0, -1.D0, 1.D0,
  T(1)=0.D0, TMAX=.1D2,
  YZERO(1)=1.D0, A(1,1)=-1.D2,
  H=1.25D0,
  IGFN=1, IPC=0, METHOD=1, INDEX=1,
 $END
```

Variable step size, LMS of order 3.

```
 $INPUTS
  N=1, KSTEP=3, ALPHA(1)=0.D0, 0.D0, -1.D0, 1.D0,
  T(1)= 0D0, TMAX=.1D2,
  YZERO(1)=1.D0, ERR=.1D-11,
  H=.1D-3,  HMAX=.1D-2,
  IPC=1, METHOD=0, INDEX=0
 $END
```

User-Supplied Subroutines

```
      SUBROUTINE GFN (G,H,N,Y,J,T,A,NR,NC)
      IMPLICIT REAL*8 (A-H, O-Z)
      DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)
      TH = T (1) + (J-1) *H
      G (1) = 1. + TH * TH
      RETURN
      END
```

User-Supplied Subroutines (cont)

```
SUBROUTINE FFN (Y,N,FN,I)
PARAMETER  NR=20, NC=20
IMPLICIT REAL*8 (A-H, O-Z)
COMMON A (NR, NC), ALPHA (4), T (NC)
DIMENSION FN (1), Y (4, NC)
FN (1) = -100, *Y (I, 1) + (1.+T (I) **2)
RETURN
END
```

An Exact Starter - BEGIN

```
SUBROUTINE BEGIN (K,H,N,Y,YN,YOLD,INVERT,IT,METHOD, IAT)
PARAMETER NR=20, NC=20
IMPLICIT REAL*8 (A-H, O-Z)
EXTERNAL INVERT
COMMON Y (4, NC), YN (N), YOLD (4, NC)
H2 = T (1) + H
H3 = H2 + H
H4 = H3 + H
X = 1.-.01-2./100.**3
X4=X*DEXP(-100.*H4)
X3=X*DEXP(-100.*H3)
X=X*DEXP(-100.*H2)
W=(100.*H2)**2-200.*H2+2.
Y(2,1)=X+.01+W/100.**3
W=(100.*H3)**2-200.*H3+2.
Y(3,1)=X3+.01+W/100.**3
W=(100.*H4)**2-200.*H4+2.
Y(4,1)=X4+.01+W/100.**3
RETURN
END
```

SOURCE: Lee[1].

Numerical Results

Give results around t = 5, 10.

Implicit NLMS methods of order 3.

t	**y**(t)
.50000000+01	.25900200-00*
	.25900200-00†
.10000000+02	.10080020+01*
	.10080020+01†

LMS methods of order 3

t	**y**(t)
.50047500+01	.25497628-00*
	.25497628-00†
.10020750+02	.10121522+01*
	.10121522+01†

Remarks

Nonlinear multistep methods can arrive at t = 10 in one step by using h = 2.5 at the cost of 4.1 seconds CPU time. However, in order to obtain **y** (5), we use h = 1.25; the CPU time costs a little more—4.4 seconds. These NLMS methods are designed to allow the use of large step size and to be effective for those **g**(t, **y**) belonging to the class of low-order polynomials. Therefore, it is not surprising that the implicit NLMS method of order 3 produces accurate results.

*Solution produced by the methods
†Exact solution

Problem 2

$$\mathbf{y}' = \begin{pmatrix} \frac{\delta k - \beta}{\ell} & \lambda_1 \\ \frac{\beta_1}{\ell} & -\lambda_1 \end{pmatrix} \mathbf{y} \; ; \; \mathbf{y}(0) = \begin{pmatrix} 1 \\ -1 \end{pmatrix} ,$$

using $\delta k = 1.0075$

$\beta = \beta_1 = 0.0075$

$\lambda_1 = 0.075$

$\ell = 10^{-6}$

Exact Solution

$\mathbf{y}(2) = (-.6414\ 1073\text{-}07,\ -.8552\ 1424 + 00)^T$

$\mathbf{y}(5) = (-.5130\ 4190\text{-}07,\ -.6840\ 5581 + 00)^T$

$\mathbf{y}(10) = (-.3536\ 0130\text{-}07,\ -.4714\ 6837 + 00)^T$

Classificiation

(0, 1, 0, 0)

Eigenvalues of $\mathbf{A}$

$\{-.0744375,\ -10^6\}$

Method Applied

Implicit NLMS of order 2 with fixed step size.

Step Size Used

1.0

Time Interval

$[0,\ 10]$

Special Features Exercised

Fixed step size
Exact Solution

FORTRAN V Inputs Setup

Fixed step size, implicit NLMS of order 2

```
 $INPUTS
  N = 2, KSTEP = 2, ALPHA (1) = 0.D0, -1.D0, 1.D0
  T (1) = 0.D0, TMAX = 10.D0
  YZERO (1) = 1.D0, -1.D0, H=1.D0
  IGFN = 0, IPC = 0, METHOD = 1, INDEX = 1
 $END
```

User-Supplied Subroutine

Since $\mathbf{g}(t, \mathbf{y}) = \mathbf{0}$, IGFN is set to 0. Hence, the GFN is not called for.

Source Weaver[10]

The infinite-medium reactor kinetic equations, in standard form, with m delayed neutron groups, can be expressed as

$$\frac{dn}{dt} = \frac{\delta k - \beta}{\ell} n + \sum_{i=0}^{m} \lambda_i C_i$$

$$\frac{dC_i}{dt} = \frac{\beta_i}{\ell} n - \lambda_i C_i,$$

where

β_i = the fraction of the total number of fission neutrons belonging to the ith group precursor that are delayed,

β = the fraction of the fission neutrons that are delayed,

ℓ = neutron generation time (sec).

λ_i = decay constant for the i th group precursor,

n = neutron density,

C_i = the concentration of the i th precursor, and

δ_k = a constant for a reactor with a constant excess of reactivity.

Considering a single delayed neutron group, we obtain

$$\begin{pmatrix} \frac{dn}{dt} \\ \\ \frac{dC_1}{dt} \end{pmatrix} - \begin{pmatrix} \frac{\delta k - \beta}{\ell} & \lambda_1 \\ \\ \frac{\beta}{\ell} & -\lambda_1 \end{pmatrix} \begin{pmatrix} n \\ \\ C_1 \end{pmatrix}$$

Numerical Results

Give results at t = 2, 5, 10.

t	y_1 (t)	y_2(t)
.2000 0000 + 01	-.6414 1073-07	-.8552 1424+00*
	-.6414 1073-07	-.8552 1424+00†
.5000 0000+01	-.5130 4190-07	-.6840 5581+00
	-.5130 4190-07	-.6840 5581+00
.1000 0000+02	-.3536 0130-07	-.4714 6837+00
	-.3536 0130-07	-.4714 6837+00

Remarks

A-stable NLMS methods can solve homogeneous stiff equations exactly in the absence of round-off errors. An extremely small step size is required for the same accuracy (.1 x 10^{-6}) if LMS methods are used.

*Produced by implicit NLMS methods of order 2
†Exact solution

Problem 3

$$\mathbf{y}' = \begin{pmatrix} 0 & 1 \\ 10 & -9 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} \; ; \quad \mathbf{y}(0) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} ,$$

Exact Solution

$$\mathbf{y}(t) = \begin{pmatrix} 2e^{t}-1 \\ 2e^{t}-1 \end{pmatrix}$$

Classification

$$(0, 0, 1, 0)$$

Eigenvalues of $\mathbf{A}$

$$\{-10, 1\} \; .$$

Methods Applied

Explicit NLMS method of order 3.

Step Size Used

0.1 and 1.0.

Time Interval

$$[0, 10] \; .$$

Special Features Exercised

Fixed step size.

Self-start.

FORTRAN V Inputs Set-Up

Fixed step size, explicit NLMS of order 3.

```
 $INPUTS
  N=2, KSTEP=3, ALPHA (1)=0.D0, 0.D0, -1.D0, 1.D0,
  T(1)=0.D0, TMAX=.1D2,
  YZERO(1)=1.D0, 1.D0,
  A(1,1)=0.D0, A(1,2)=1.D0, A(2,1)=.1D2, A(2,2)=-9.D0,
  H=1.D0,
  IGFN=1, IPC=0, METHOD=1, INDEX=0
 $END
```

User-Supplied Subroutines

```
   SUBROUTINE GFN (G,H,N,Y,J,T,A,NR,NC)
   IMPLICIT REAL*8 (A-H, O-Z)
   DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)
   G (1) = 1.0
   G (2) = 1.0
   RETURN
   END
```

Source: Lee[1].

Numerical Results

Gives results at t=1, 5, and 10.

t	$y_1(t)$	$y_2(t)$
.10000000+01	.44365637+01	.44365637+01*
	.44365637+01	.44365637+01†
.50000000+01	.29582632+03	.29582632+03
	.29582632+03	.29582632+03
.10000000+02	.44051932+05	.44051932+05
	.44051932+05	.44051932+05

*Produced by explict NLMS methods of order 3.
†Exact solution.

Since LMS methods are not used, FFN is not called for. The built-in self-starter is used.

Problem 4

$u'' + u = 0;\ u(0) = 0,\ u'(0) = 1.$

Equivalent Problem

$$\begin{pmatrix} u' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} u \\ y \end{pmatrix};\quad \begin{pmatrix} u(0) \\ y(0) \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Exact Solution

$$u = \sin t.$$

Classification

$$(1,0,0,0)$$

Eigenvalues of $\mathbf{A}$

$$\{\pm i\}.$$

Method Applied

LMS methods of order 2 with PC^m procedure.

Step Size Used

Initial step size $= \pi$.

Remarks

In the absence of round-off errors, NLMS methods of different orders are equivalently accurate if $\mathbf{g}(t, \mathbf{y})$ is a constant; NLMS methods of any order should produce accurate results for this problem.

Time Interval

$$[0, 6\pi] \ .$$

Special Features Exercised

Variable step size

PC^m procedure

Exact start.

FORTRAN V Inputs Set-Up

```
 $INPUTS
  N=2, KSTEP=2, IPC=1, INDEX=0, METHOD=0
  T(1)=0.D0, TMAX=18.75D0
  YZERO(1)=0.D0, 1.D0, ERR=.1D-11
  ALPHA(1) = 0.D0, -1.D0, 1.D0
 $END
```

In MAIN, H is defined to be π initially and HMAX = H/16.

User-Supplied Subroutines

```
  SUBROUTINE FFN (Y,N,FN,I)
  COMMON A (20, 20), ALPHA (4), T (20)
  DIMENSION FN(20, Y(4,20)
  FN (1) = Y (I, 2)
  FN (2) = -Y (I, 1)
  RETURN
  END
```

Source: Krough[9]

Numerical Results

Give maximum absolute error at t = 2, 4, 6.

t	Maximum Absolute Error
2π	.2307 1215-12
4π	.2307 0955-12
6π	.2307 1128-12

Remarks

This problem demonstrates that high-order differential equations can be decomposed into a system of first-order equations so that this package is applicable. Since this is not a stiff equation, we choose the variable-step-size and PC^m procedure to examine the efficiency of the LMS portion.

Problem 5

$$\mathbf{y}' = t\,(1 - \mathbf{y}) + (1-t)e^{-t};\ \mathbf{y}\,(.1) = 1.0901751.$$

Equivalent Problem

$$\mathbf{y}' = -t\,\mathbf{y} + t + (1-t)e^{-t};\ \mathbf{y}(.1) = 1.0901751.$$

Exact Solution

$$\mathbf{y}(t) = e^{-\frac{t^2}{2}} - e^{-t} + 1.$$

Classification

$0 < t \leq 1$	$(0, 0, 1, 0)$
$1 < t$	$(0, 1, 1, 0)$.

Eigenvalues of $\mathbf{A}$

$$\{-t\}.$$

Methods Applied

NLMS methods of order 3.

Step Size Used

Variable step size with initial $h = 0.01$.

Time Interval

$[0.1, 50.0]$.

Special Features Exercised

Automatic periodic decomposition of **A** (t).

PC^m procedure.

Variable-step-size procedure.

FORTRAN V Inputs Set-Up

Variable step size, NLMS of order 3

```
 $INPUTS
  N=1, KSTEP=3, ALPHA(1)=0.D0, 0.D0, -1.D0, 1.D0,
  T(1)=.1D0, TMAX=.5D2,
  YZERO(1)=1.0901751D0, ERR=.1D-11,
  H=.1D-1, HMAX=.1D0,
  IGFN=1, IPC=1, METHOD=1, INDEX=0, IAT=1
 $END
```

User-Supplied Subroutines

```
  SUBROUTINE GFN (G,H,N,Y,J,T,A,NR,NC)
  IMPLICIT REAL*8 (A-H, O-Z)
  DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)
  G (1) = T (J) + (1.-T (J))*DEXP(-T (J))
  RETURN
  END

  SUBROUTINE AFNT (A,N,T,NR,NC)
  DOUBLE PRECISION A (NR, NC), T
  A (1,1) = -T
  RETURN
  END
```

An Exact Starter - BEGIN

```
    SUBROUTINE BEGIN (K,H,N,Y,YN,YOLD,INVERT,IT,METHOD, IAT)
    PARAMETER NR=20, NC=20
    IMPLICIT REAL*8 (A-H, O-Z)
    EXTERNAL INVERT
    COMMON A (NR, NC), ALPHA (4), T(NC)
    DIMENSION Y (4, NC), YN (N), YOLD (4, NC)
    DO 1 I = 2, 4
    T (I) = T (I-1) + H
  1 Y (I, 1) = DEXP (-T (I) * T (I)/2.) -DEXP (-T (I)) + 1.0
    RETURN
    END
```

Source: Krough[9].

Numerical Results

Give results at t = 1.0, 10, and 30.

t	y (t)
.10000000+01	.12386512-01*
	.12386512-01†
.10000000+02	.99995460+00
	.99995460+00
.30000000+02	.10000000+01
	.10000000+01

*Produced by NLMS methods of order 3 .
†Exact solution

Problem 6

$$\mathbf{y}' = -100\, t\, \mathbf{y}^2; \quad \mathbf{y}(1) = \frac{1}{51}$$

Equivalent Problem

$$\mathbf{y}' = -100\ \mathbf{y} + 100\ \mathbf{y}(1 - t\,\mathbf{y}); \quad \mathbf{y}(1) = \frac{1}{51}$$

Exact Solution

$$\mathbf{y}(t) = \frac{\quad}{1 + 50t^2}$$

Classification

(0,1,1,1)

Eigenvalues of $\mathbf{A}$

$$\{-100\} \ .$$

Methods Applied

NLMS methods of order 3.

Step Size Used

Variable step size with initial h = 0.01.

Time Interval

$$[1,\ 50] \ .$$

Special Features Exercised

Variable step size

$\mathbf{g} = \mathbf{g}(t, \mathbf{y})$,

Exact start.

PC^m procedure.

FORTRAN V Inputs Set-up

Variable step size, NLMS order of 3.

```
 $INPUTS
  N=1, KSTEP=3, ALPHA(1)=0.D0, 0.D0, -1.D0, 1.D0,
  T(1)=1.D0, TMAX=.5D2,
  YZERO(1)=.19607843D-1, ERR=.1D-11,
  A (1,1) = -100.D0,
  H = .1D-1, HMAX = .1D0,
  IGFN = 1, IPC = 1, METHOD = 1, INDEX = 0
 $END
```

User-Supplied Subroutines

```
   SUBROUTINE GFN (G,H,N,Y,J,T,A,NR,NC)
   IMPLICIT REAL *8 (A-H, O-Z)
   DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)
   G (1) = 100.*Y (J, 1) * (1.-T (J)*  Y (J, 1))
   RETURN
   END
```

An Exact Starter: BEGIN

```
   SUBROUTINE BEGIN (K,H,N,Y,YN,YOLD,INVERT,IT,METHOD,IAT)
   PARAMETER NR=20, NC=20
   IMPLICIT REAL*8 (A-H, O-Z)
   EXTERNAL INVERT
   COMMON A (NR, NC), ALPHA (4), T (NC)
   DIMENSION Y (4, NC), YN (N), YOLD (4 (NC))
   DO 1 I = 2,4
   T (I) = T (I-1) + H
 1 Y (I, 1) = 1./(1.+50.*T (I))
   RETURN
   END
```

Source: Jain[8].

Numerical Results

Give results around t = 5, 10, 20, 30, and 50.

.5003 1250 +01	.7983 6304-03*
	.7983 6304-03†
.1000 1875 +02	.19988506-03
	.19988506-03
.2000 6875 +02	.49963146-04
	.49963146-04
.3000 4375 +02	.22215249-04
	.22215249-04
.5001 4375 +02	.7995 3381-05
	.7995 3381-05

Remarks

The decomposition technique is applied successfully with the PC^m procedure for this nonlinear problem. Here, $\mathbf{g}(t, \mathbf{y})$ is not a low-order polynomial, but the program can be seen to vary the step size automatically and to maintain convergence and reasonable accuracy.

Problem 7

$$\mathbf{y}' = \mathbf{U}\mathbf{Z} - \mathbf{U}\mathbf{B}\mathbf{U}\mathbf{y} \quad ; \quad \mathbf{y}(0) = (-1, -1, -1, -1)^T$$

where

$$\mathbf{U} = \frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$$

$$\mathbf{Z} = \left(w_1^2, w_2^2, w_3^2, w_4^2\right)^T$$

*Solution produced by NLMS methods
†Exact solution.

$$\mathbf{W} = \left(w_1, w_2, w_3, w_4\right)^T = \mathbf{U}\mathbf{y}$$

$$\mathbf{B} = \text{diag}\,(\beta_1, \beta_2, \beta_3, \beta_4)$$

$$\beta_1 = 1000,\ \beta_2 = 800,\ \beta_3 = -10,\ \beta_4 = 0.001.$$

Exact Solution

$$\mathbf{y}(t) = \mathbf{U}\left(\frac{\beta_1}{1-(1+\beta_1)e^{\beta_1 t}},\ \frac{\beta_2}{1-(1+\beta_2)e^{\beta_2 t}},\ \frac{\beta_3}{1-(1+\beta_3)e^{\beta_3 t}},\ \frac{\beta_4}{1-(1+\beta_4)e^{\beta_4 t}}\right)^T.$$

Classification

(0, 1, 1, 0).

Eigenvalues of $\mathbf{A}$

$\{-1002, -802, 8, -2.001\}$ initially

$\{-1000, -800, -10, -0.001\}$ when $0.001t \geq 1$.

Method Applied

NLMS methods of order 1

Step Size Used

$0.01 \leq t \leq 1$, h = .1D-2.

$1 < t \leq 10$, h = .1D-1,

$10 < t \leq 1000$, h = .1D0.

Time Interval

$[0.01, 100]$.

Special Features

Fixed step size.

FORTRAN V Inputs Set-Up

To maintain better precision, matrix **A** (A(I,J)) and vector $\mathbf{y}_0$ (YZERO) are calculated by programs specially incorporated in MAIN.

Fixed step size, NLMS of order 1.

```
 $INPUTS
  N = 4, KSTEP  = 1, ALPHA (1) = -.1D0, 1.D0,
  T (1) = .1D-1, TMAX = 1.D3
  H = .1D-2,
  IGFN = 1, IPC = 0, METHOD = 1, INDEX = 0
 $END
```

User-Supplied Subroutines

```
   SUBROUTINE GFN (G,H,N,Y,J,T,A,NR,NC)
   IMPLICIT REA*8 (A-H, O-Z)
   DIMENSION A (NR, NC), Y (4, NC), G (NC), T (NC)
   DIMENSION W (4)
   WW = 0.0
   D0 10 K =1,N
10 WW = WW + Y (J,K)
   D0 5 I = 1, N
   W (I) = WW -2.*Y (J, I)
 5 W (I) = W (I) *W (I) /4.0
   G (1) = (-W(1) + W(2) + W(3) + W(4))/2.0
   G (2) = ( W(1) - W(2) + W(3) + W(4))/2.0
   G (3) = ( W(1) + W(2) - W(3) + W(4))/2.0
   G (4) = ( W(1) + W(2) + W(3) - W(4))/2.0
   RETURN
   END
```

Source: Krough, Gear[7].

Numerical Results

Given results at t = 50, 500, 1000

t				
50	-.50095236+01	-.50095326+01	.49904764+01	-.49904764+01*
	-.50095561+01	-.50095561+01	.49904439+01	-.49904439+01†
500	-.50007681+01	-.50007681+01	.49992319+01	-.49992319+01
	-.50007688+01	-.50007688+01	.49992312+01	-.49992312+01
1000	-.50002903+01	-.50002903+01	.49997097+01	-.49997097+01
	-.50002905+01	-.50002905+01	.49997095+01	-.49997095+01

Remarks

Since only three different step sizes are used, there are only three matrix inversions and three matrix exponentials needed. To reach t = 1000 takes about 4 min 46 sec with a maximum error $\sim .38 \times 10^{-7}$.

*Solution produced by NLMS methods.
†Exact solution.

9. CONCLUSIONS

A family of strongly stable and consistent NLMS methods has been developed. When applying NLMS methods to the problem $\mathbf{y} = \mathbf{A}\mathbf{y}$, which implies $\mathbf{g}(t, \mathbf{y}) = \mathbf{0}$, NLMS methods produce the approximate solution $\mathbf{y}_n = e^{\mathbf{A}t_n}\,\mathbf{y}_o = e^{n\mathbf{A}h}\,\mathbf{y}_0$. Since the matrix exponential is computed by the Pade approximation,[12,1] which is stable, the lim $\|\mathbf{y}_n\| \longrightarrow 0$ as $n \to \infty$ establishing the A-stability in the sense of Dahlquist.[13]

If $\mathbf{g}(t, \mathbf{y})$ is a slowly varying function that can be approximated by a low-order polynomial in t and $\mathbf{A}$ is a slowly varying function in t, NLMS methods allow the use of a larger step size and can produce almost exact solution in the absence of round-off errors. In the event that $\mathbf{g}(t, \mathbf{y})$ is not a low-order polynomial in t or not a slowly varying function in t, NLMS methods can still produce accurate results if not too large a step size is used.

In general, because of the initial local discretization error, implicit NLMS methods are better than explict NLMS methods. High-order NLMS methods are better than low-order NLMS methods. In the event that $\mathbf{g}(t, \mathbf{y})$ is a constant in t and $\mathbf{y}$, we see that

$$\sum_{i=0}^{K-1} \phi_{Ki}^{(E)} = \sum_{i=0}^{K} \phi_{Ki}^{(I)} .$$

Therefore, explicit NLMS of order K is equivalent to implicit NLMS of order K. In this case, low-order NLMS methods will suffice; high-order NLMS methods are not necessary. In fact, high-order NLMS methods require more computations, a good Pade approximation, and an accurate matrix inversion.

Several numerical experiments has been performed upon test problems utilizing different selection of α_i, from both strongly and weakly stable NLMS families. The numerical results agree to nine significant digits.

The numerical results presented in this report mainly are used to demonstrate that the NLMS program can produce satisfactory results. The computations herein were intended for illustrative purposes only. It is our intention to apply NLMS methods to more difficult problems. We intend to develop criteria to determine how good NLMS methods are. These will be reported separately.

The NLMS package requires the user to guess an initial step size h. However, the package is "robust"; it determines the most favorable step size for rapid computation. Experience indicates that less total computation time results if h is chosen too small, rather than too large.

If a large, stiff system is to be solved with the variable-step-size technique, an eigenvalue-eigenvector technique such as EISPACK is recommended to replace PADE. This can save a large amount of computation time.

10. REFERENCES

1. D. Lee, Nonlinear Multistep Methods for Solving Initial Value Problems in Ordinary Differential Equations, Ph. D. dissertation, Polytechnic Institute of New York, 1974. Also NUSC Technical Report 4775, 24 May 1974.

2. P. Henrici, Discrete Variable Methods in Ordinary Differential Equations, John Wiley and Sons, Inc., New York, 1962.

3. H.B. Keller, Numerical Methods for Two-Point Boundary-Value Problems, Blaisdell Publishing Co., Waltham, Massachusetts, 1968.

4. J. Certaine, "The Solution of Ordinary Differential Equations With Large Time Constants," in Mathematical Methods for Digital Computers, A. Ralston and H. Wilf, eds., John Wiley and Sons, Inc., New York, 1960, pp. 128-132.

5. L.F. Shampine and M.K. Gordon, Computer Solution of Ordinary Differential Equations, W.H. Freeman and Co., San Francisco, 1975.

6. B.L. Ehle, A Comparison of Numerical Methods for Solving Certain Stiff Ordinary Differential Equations, Department of Mathematics, University of Victoria, Canada, Report 70, 1972.

7. C.W. Gear, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

8. R.K. Jain, "Some A-Stable Methods for Stiff Ordinary Differential Equations," Mathematics of Computation, vol. 26, no. 117, pp. 71-77, 1972.

9. F.T. Krough, "On Testing a Subroutine for the Numerical Integration of Ordinary Differential Equations," Journal of the ACM, vol. 20, pp. 545-562, 1973.

10. L.E. Weaver, Systems Analysis of Nuclear Reactor Dynamics, Rowman and Littlefield, Inc., New York, 1963.

11. R.A. Willoughby, Stiff Differential Systems, Plenum Press, New York and London, 1974.

12. J.L. Blue, and H.K, Gummel, "Rational Approximations to Matrix Exponential for Systems of Stiff Equations," Journal of Computational Physics, vol. 5, 1970, pp. 70-83.

13. G. Dahlquist, "A Special Stability Problem for Linear Multistep Methods," BIT, vol. 3, 1963, pp. 27-43.

Appendix

NLMS COMPUTER PROGRAMS

```
C*************************************************************************
C*   THIS PROGRAM COMMANDS A SET OF SUBROUTINES TO INTEGRATE A           *
C*   SYSTEM OF FIRST-ORDER ORDINARY DIFFERENTIAL EQUATIONS OF            *
C*   TH FORM                                                             *
C*                        DY(T)/DT=AY + G(T,Y)                           *
C*                  OR    DY(T)/DY=F(T,Y)                                *
C*                                                                       *
C*   OVER THE CLOSED INTERVAL-(T(1),TMAX). THIS IS A FIXED-ORDERING,     *
C*   VARIABLE(FIXED AS WELL)-STEP-SIZE PROGRAM. MOST INPUTS ARE          *
C*   DETECTED BY ASSIGNING REASONABLE DEFAULT VALUES. ONLY DIFEQ IS      *
C*   CALLED BY THIS PROGRAM. OTHER SUBROUTINES ARE CALLED BY             *
C*   SUBROUTINES. UNUSED SUBROUTINES ARE DEFINED DUMMIES TO SATISFY      *
C*   THE COMPILER. ALL COMPUTATIONS ARE PERFORMED IN DOUBLE-PRECISION.   *
C*   INPUT PARAMETERS APPEAR ON THE NAMELIST HAVE THE FOLLOWING          *
C*   DEFINITIONS                                                         *
C*                                                                       *
C*   A         A 2-DIMENSIONAL ARRAY OF (NR X NC) FOR STORING            *
C*             ELEMENTS OF MATRIX A                                      *
C*                                                                       *
C*   ALPHA     CHARACTERISTIC POLYNOMIAL COEFFICIENTS. NO CONCERN TO     *
C*             THE USER. PROGRAM SELECTS ADAMS COEFFICIENTS FOR          *
C*             STRONG STABILITY                                          *
C*                                                                       *
C*   ERR       USER-SUPPLIED TOLERANCE. DEFAULT VALUE 0.11D-11           *
C*                                                                       *
C*   H         USER-SUGGESTED INITIAL STEP SIZE. DEFAULT VALUE 0.001     *
C*                                                                       *
C*   HMAX      THE MAXIMUM STEP SIZE THE USER ALLOWS THE PROGRAM TO      *
C*             CONSIDER                                                  *
C*                                                                       *
C*   IAT       SET =0, IF A IS A CONSTANT                                *
C*             SET =1, IF A IS A FUNCTION OF T                           *
C*                                                                       *
C*   IGFT      SET =0, IF G(T,Y) IS ABSENT                               *
C*             SET =1, IF G(T,Y) IS PRESENT                              *
C*                                                                       *
```

```
C*   INDEX    SET =0 FOR EXPLICIT. SET =1 FOR IMPLICIT                     *
C*                                                                         *
C*   IPC      SET =0 FOR PREDICTOR OR CORRECTOR                            *
C*            SET =1 FOR PREDICTOR-AND-CORRECTOR                           *
C*                                                                         *
C*   KSTEP    STEP NUMBER (LESS THAN 4)                                    *
C*                                                                         *
C*   METHOD   SET =0, LMS  METHODS                                         *
C*            SET =1, NLMS METHODS                                         *
C*                                                                         *
C*   N        NUMBER OF EQUATIONS                                          *
C*                                                                         *
C*   T        A 1-DIMENSIONAL ARRAY. T(1) CONTAINS INITIAL TIME            *
C*                                                                         *
C*   TMAX     FINAL TIME, ASSIGNED BY THE USER                             *
C*                                                                         *
C*   YZERO    A 1-DIMENSIONAL ARRAY, CONTAINING INITIAL Y                  *
C*                                                                         *
C*   START    A SELF-START SUBROUTINE                                      *
C*   BEGIN    USER-SUPPLIED STARTER (OPTIONAL)                             *
C*   GJR      A BUILT-IN MATRIX INVERSION SUBROUTINE                       *
C*   INVERT   A USER-SUPPLIED MATRIX INVERSION (OPTIONAL)                  *
C*   START,BEGIN,GJR,INVERT ALL ENTER ARGUMENTS OF CALL DIFEQ              *
C*                                                                         *
C*   INPUT SET UP AS FOLLOWS--                                             *
C*                                                                         *
C*            $INPUTS                                                      *
C*                USERS, INPUT DATA                                        *
C*            $END                                                         *
C***************************************************************************
      PARAMETER NR=20, NC=20
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL START,BEGIN,INVERT
      COMMON A(NR,NC),ALPHA(4),T(NC)
      DIMENSION Y(4,NC),YZERO(NC)
      DATA H,HMAX,ALPHA/.1D-2,.100,4*1.D3/
      DATA KSTEP,METHOD,INDEX/2*1,0/
      NAMELIST /INPUTS/A,ALPHA,ERR,H,HMAX,IAT,IGFN,INDEX,IPC,KSTEP,
     *                    METHOD,N,T,TMAX,YZERO
   10 READ(3,INPUTS,ERR=100,END=101)
      CALL DIFEQ(ERR,H,HMAX,KSTEP,N,TMAX,Y,YZERO,BEGIN,INVERT,IGFN,
     *              IPC,METHOD,INDEX,IAT)
      GO TO 10
  100 STOP 100
  101 STOP 101
      END


      SUBROUTINE DIFEQ(ERR,H,HMAX,KSTEP,N,TMAX,Y,YZERO,START,INVERT,
     *                     IG,IV,METHOD,INDEX,IAT)
C*********************************************************************
C*   DIFEQ IS CALLED BY MAIN PROGRAM WHOSE ARGUMENTS ARE ALREADY     *
C*   DEFINED IN THE MAIN PROGRAM WHERE IG=IGFN, IV=IPC               *
C*                                                                   *
C*   DIFEQ CALLS 4 SUBROUTINES                                       *
C*         START(KSTEP,...,IAT) --- A STARTER                        *
C*         LMS(KSTEP,...,I)     --- LINEAR MULTISTEP                 *
C*         NLMS(KSTEP,...,IAT)  --- NONLINEAR MULTISTEP              *
C*         PRINT(TEA,YNEW)      --- FOR USER TO PRINTOUT RESULTS     *
C*                                                                   *
C*   SOLUTION VECTOR Y(T) IS YNEW(I) OR Y(KSTEP+1,I)                 *
C*                                                                   *
C*   BASED ON USER-SUPPLIED INPUTS, DIFEQ SETS UP THE ITERATIVE      *
C*   PROCEDURE, CONTROLS STARTER, STEP-SIZE CHANGES, PREDICTOR-      *
C*   CORRECTOR, CORRECTOR,S CONVERGENCE, PRINTOUTS AND CALLS FOR     *
C*   THE REQUIRED METHODS                                            *
C*********************************************************************
      PARAMETER NR=20, NC=20
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL INVERT,START
      COMMON A(NR,NC),ALPHA(4),T(NC)
      DIMENSION G(NC),Y(4,NC),YN(NC),YNEW(NC),YOLD(4,NC),YZERO(NC)
      DATA HMIN/.1D-11/
      LMT=3
      WRITE (4,10)
   10 FORMAT(1H1)
C***********************************************************************
C*   SELECTS ADAMS ALPHA IF ALPHA ARE NOT GIVEN                        *
C***********************************************************************
      IF(ALPHA(1) - 10.0) 37,,
      DO 36 I=1,KSTEP
   36 ALPHA(I)=0.0
      ALPHA(KSTEP)=-1.0
      ALPHA(KSTEP+1)=1.0
   37 DO 40 I=1,N
   40 Y(1,I)=YZERO(I)
      ITER=0
      ISTEP=0
      TZERO=T(1)
   49 ITER=ITER+1
      IH=0
      IMIN=0
   50 CONTINUE
C***********************************************************************
C*   EVALUATE A(T) AT T=T(0) IF IAT NOT 0                              *
C*   IF KSTEP GT 1, CALLS START                                        *
C***********************************************************************
      IF(IAT.NE.0) CALL AFNT(A,N,T(1),NR,NC)
      IF(KSTEP.EQ.1 ,AND. INDEX.EQ.0) GO TO 60
      CALL START(KSTEP,H,N,Y,YN,YOLD,INVERT,IG,METHOD,IAT)
   60 TEA=T(1)+KSTEP*H
      IH=IH+1
      DO 61 J=1,KSTEP
   61 T(J+1)=T(J)+H
```

```
      DO 62 J=0,KSTEP
      DO 62 I=1,N
   62 YOLD(J+1,I)=Y(J+1,I)
C***********************************************************************
C*   METHOD=0 -- LMS, METHOD=1 -- NLMS                                 *
C*   FIXED-STEP SIZE         IV=0, INDEX=0  PREDICTOR                  *
C*                           IV=0, INDEX=1  CORRECTOR                  *
C*   VARIABLE-STEP-SIZE      IV NOT 0       PREDICTOR $ CORRECTOR      *
C***********************************************************************
      IF(METHOD.NE.0) GO TO 64
      IF(INDEX.EQ.0) CALL LMS(KSTEP,H,YOLD,N,YN,0)
      IF(IV.EQ.0.AND,INDEX.EQ.1) CALL LMS(KSTEP,H,YOLD,N,YN,1)
      GO TO 59
   64 IF(IV.EQ.0.AND,INDEX.EQ.0) CALL NLMS(KSTEP,H,YOLD,N,YN,0,IH,
     $                                    INVERT,IG,IAT)
      IF(IV.EQ.0.AND,INDEX.NE.0) CALL NLMS(KSTEP,H,YOLD,N,YN,1,IH,
     $                                    INVERT,IG,IAT)
      IF(IV.NE.0) CALL NLMS(KSTEP,H,YOLD,N,YN,0,1,INVERT,IG,IAT)
   59 DO 66 I=1,N
   66 YOLD(KSTEP+1,I)=YN(I)
      IF(IV.NE.0 .OR, IAT.NE.0) GO TO 69
      IF(LMT .EQ. 1) GO TO 69
      DO 68 I=1,N
   68 YNEW(I)=YN(I)
      GO TO 82
C***********************************************************************
C*   CORRECTOR AT MOST CORRECT 3 TIMES                                 *
C*   STEP-SIZE CHANGED BY A FACTOR OF 2                                *
C***********************************************************************
   69 ICORR=0
   70 ICORR=ICORR+1
      IF(ICORR .LE. LMT) GO TO 75
      H=H/2.0
      IF(H .LT. HMIN) IMIN=IMIN+1
      IF(H .LT. HMIN) H=HMIN
      IF(IMIN .GT. 3) WRITE (4,1170)
```

```
 1170 FORMAT(3X,'H REACHED HMIN, NO CONVERGENCE POSSIBLE')
      IF(IMIN .GT. 3) STOP
      T(1)=TZERO
      DO 71 I=1,N
   71 Y(1,I)=YZERO(I)
      GO TO 50
   75 IF(METHOD.EQ.0) CALL LMS(KSTEP,H,YOLD,N,YNEW,1)
      IF(METHOD .GT. 0) CALL NLMS(KSTEP,H,YOLD,N,YNEW,1,ICORR,INVERT,
     *IG,IAT)
      IF(LMT .NE. 1) GO TO 76
      DO 77 I=1,N
   77 YNEW(I)=YN(I)
      GO TO 82
   76 CONTINUE
      ANORM=0.0
C***********************************************************************
C*   TEST CORRECTOR,S CONVERGENCE, USING MAX NORM                      *
C***********************************************************************
      DO 80 J=1,N
      G(J)=(YOLD(KSTEP+1,J)-YNEW(J))/YOLD(KSTEP+1,J)
      IF(ANORM .GT. DABS(G(J))) GO TO 80
      ANORM=DABS(G(J))
   80 CONTINUE
      IF(ANORM - ERR) 82,82,
      DO 81 J=1,N
   81 YOLD(KSTEP+1,J)=YNEW(J)
      GO TO 70
   82 DO 83 I=1,N
   83 Y(KSTEP+1,I)=YNEW(I)
C***********************************************************************
C*   RESULTS Y(TEA) IN YNEW(I) AND Y(KSTEP+1,I)                        *
C*   CALL PRINT FOR USER TO PRINTOUT RESULTS                           *
C***********************************************************************
      CALL PRINT(H,TEA,YNEW,N,ITER)
   84 CONTINUE
```

```
      IF(TEA .GT. TMAX) RETURN
      DO 85 J=1,KSTEP
      DO 85 I=1,N
      Y(J,I)=Y(J+1,I)
      IF(J.EQ.1) YZERO(I)=Y(J,I)
   85 CONTINUE
      T(1)=T(2)
      TZERO=T(1)
      IF(IV.EQ.0 .AND. INDEX.EQ.1) LMT=1
      IF(IV.EQ.0 .AND. INDEX.EQ.1) INDEX=0
      IF(LMT .EQ. 1) IH=0
      IF(IV .EQ. 0) GO TO 60
      T(1)=TEA
      TZERO=T(1)
      DO 86 I=1,N
      YZERO(I)=Y(KSTEP+1,I)
   86 Y(1,I)=Y(KSTEP+1,I)
C***********************************************************************
C*   MAINTAIN SUCCESSFUL H CONSTANTLY FOR 3 TIMES BEFORE DOUBLING      *
C***********************************************************************
      ISTEP=ISTEP+1
      IF(ISTEP .LT. 3) GO TO 49
      ISTEP=0
      H=2.*H
      IF(H .GT. HMAX) H=HMAX
      GO TO 49
      END

      SUBROUTINE START(KSTEP,H,N,Y,YN,YOLD,INVERT,IT,METHOD,IAT)
C***********************************************************************
C*   A SELF-STARTER, CALLED BY DIFEQ                                   *
C*   ARGUMENTS ALREADY DEFINED IN MAIN PROGRAM                         *
C*   FINAL VALUES ARE STORED IN Y(J,I)                                 *
C*   THIS PROGRAM CALLS 2 SUBROUTINES                                  *
C*           LMS(1,...,0)                                              *
C*           NLMS(1,...,IAT)                                           *
C***********************************************************************
```

```
      PARAMETER NR=20, NC=20
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL INVERT
      COMMON A(NR,NC),ALPHA(4),T(NC)
      DIMENSION Y(4,NC),YN(N),YOLD(4,NC)
      AL=ALPHA(1)
      ALPHA(1)=-1.0
      DO 1 I=1,N
    1 YOLD(1,I)=Y(1,I)
      DO 10 J=1,KSTEP
      IF(METHOD .EQ. 1) GO TO 4
C**********************************************************************
C*   FIRST-ORDER ADAMS-BASHFORTH METHOD TO START                      *
C**********************************************************************
      HA=H/4.
      DO 3 I=1,4
      CALL LMS(1,HA,YOLD,N,YN,0)
      DO 2 L=1,N
    2 YOLD(1,L)=YN(L)
    3 CONTINUE
      GO TO 6
C**********************************************************************
C*   FIRST-ORDER GENERALIZED-ADAMS-BASHFORTH METHOD TO START          *
C**********************************************************************
    4 CALL NLMS(1,H,YOLD,N,YN,G,1,INVERT,IT,IAT)
    6 DO 5 I=1,N
      Y(J+1,I)=YN(I)
    5 YOLD(1,I)=YN(I)
   10 CONTINUE
      ALPHA(1)=AL
      RETURN
      END
      SUBROUTINE NLMS(KSTEP,H,Y,N,YN,INDEX,IS,INVERT,IT,IAT)
C**********************************************************************
C*   NONLINEAR MULTISTEP ALGORITHM(NLMS), CALLED BY DIFEQ OR START    *
C*   ARGUMENTS ALREADY DEFINED IN MAIN PROGRAM                        *
C*   THIS PROGRAM CALLS 3 SUBROUTINES                                 *
C*           INVERT(AH,...,P1)                                        *
C*           GFN(G,...,NC)                                            *
C*           PADE(A,...,INVERT)                                       *
C*   SOLUTION VECTOR IS STORED IN YN(I)                               *
C**********************************************************************
      PARAMETER NR=20, NC=20
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL INVERT
      COMMON A(NR,NC),ALPHA(4),T(NC)
      DIMENSION AH(NR,NC),AH2(NR,NC),AH3(NR,NC),AH4(NR,NC)
      DIMENSION EAH(NR,NC),E2AH(NR,NC),E3AH(NR,NC),G(NC)
      DIMENSION P(NR,NC),P1(NR,NC),PHI(4,NR,NC),QHI(4,NR,NC)
      DIMENSION UNIT(NR,NC),W(4,4,NR,NC)
      DIMENSION AT(NR,NC),Y(4,NC),YN(NC)
C**********************************************************************
C*   IS IS AN INDICATOR WHEN IT IS 1, PROGRAM DOES INITIALIZATION,    *
C*   CALCULATES AND SAVES AH, EXP(AH), A INVERSE, PHI FUNCTION        *
C*   IS.GT.1 ABOVE CALCULATIONS ARE BYPASSED                          *
C**********************************************************************
      IF(IS .GT. 1) GO TO (100,200,300), KSTEP
      DO 1 I=1,N
      DO 2 J=1,N
      P1(I,J)=0.0
      UNIT(I,J)=0.0
      EAH(I,J)=0.0
      E2AH(I,J)=0.0
      E3AH(I,J)=0.0
      AH2(I,J)=0.0
      AH3(I,J)=0.0
    2 AH4(I,J)=0.0
    1 UNIT(I,I)=1.0
```

```
      DO 3 I=1,4
      DO 3 J=1,N
      DO 3 K=1,N
      PHI(1,J,K)=0.0
    3 WHI(1,J,K)=0.0
      DO 6 I=1,N
      DO 6 J=1,N
    6 AH(I,J)=H*A(I,J)
      GO TO (100,200,300) , KSTEP
  100 CONTINUE
C***********************************************************************
C*   NONLINEAR MULTI-1-STEP STARTS HERE. BEGINNING SECTION DOES        *
C*   INITIALIZATION                                                    *
C***********************************************************************
      DO 132 I=1,N
      DO 133 J=1,N
  133 P1(I,J)=0.0
      YN(I)=0.0
      PHI(2,I,1)=0.0
  132 PHI(3,I,1)=0.0
      IF(IS.GT.1 .AND. INDEX.EQ.0) GO TO 131
      IF(IS.GT.1 .AND. INDEX.EQ.1) GO TO 170
C***********************************************************************
C*   P1 CONTAINS (AH)**(-1), EAH CONTAINS EXP(AH)                      *
C***********************************************************************
      IF(N-1) 120,,120
      P1(1,1)=1./AH(1,1)
      GO TO 122
  120 CALL INVERT(AH,N,P1)
  122 IF(N-1) 123,,123
      EAH(1,1)=DEXP(A(1,1)*H)
      GO TO 125
  123 CALL PADE(A,H,EAH,E2AH,E3AH,G,N,NR,NC,INVERT)
  125 DO 103 I=1,N
      DO 103 J=1,N
  103 AH(I,J)=ALPHA(1)*EAH(I,J)+UNIT(I,J)
```

```
C***********************************************************************
C*    EXPLICIT NLM-1-STEP METHODS                                      *
C*    DO LOOP 105 CALCULATES PHI(1,0)                                  *
C*    LOOP 108 OR 110 COMPUTES FINAL Y(N+1)                            *
C***********************************************************************
      IF(INDEX .NE. 0) GO TO 150
      IF(IT .EQ. 0) GO TO 109
      DO 105 I=1,N
      DO 105 J=1,N
      DO 105 K=1,N
  105 PHI(1,I,J)=PHI(1,I,J)-P1(I,K)*AH(K,J)
  131 CALL GFN(G,H,N,Y,KSTEP,T,A,NR,NC)
      DO 108 I=1,N
      DO 108 J=1,N
  108 YN(I)=YN(I)-ALPHA(1)*EAH(I,J)*Y(1,J)+H*PHI(1,I,J)*G(J)
      RETURN
  109 DO 110 I=1,N
      DO 110 J=1,N
  110 YN(I)=YN(I)-ALPHA(1)*EAH(I,J)*Y(1,J)
      RETURN
C***********************************************************************
C*    IMPLICIT NLM-1-STEP METHODS                                      *
C*    COMPUTE PHI(1,0), PHI(1,1)                                       *
C*    FINAL RESULTS ARE CALCULATED IN LOOP 166 OR 168                  *
C***********************************************************************
  150 IF(N-1) 152,,152
      AH2(1,1)=P1(1,1)*P1(1,1)
      GO TO 153
  152 DO 154 I=1,N
      DO 154 J=1,N
      DO 154 K=1,N
  154 AH2(I,J)=AH2(I,J)+P1(I,K)*P1(K,J)
  153 IF(IT .EQ. 0) GO TO 167
      DO 160 I=1,N
      DO 161 J=1,N
      DO 162 K=1,N
  162 PHI(1,I,J)=PHI(1,I,J)+ALPHA(1)*H*A(I,K)*EAH(K,J)
      PHI(1,I,J)=PHI(1,I,J)-AH(I,J)
  161 E2AH(I,J)=AH(I,J)+H*A(I,J)
  160 CONTINUE
  170 K2=KSTEP+1
      IF(IT .EQ. 0) GO TO 167
      DO 163 I=1,N
      DO 164 K=1,K2
      CALL GFN(G,H,N,Y,K,T,A,NR,NC)
      DO 165 J=1,N
      IF(K .EQ. 1) PHI(3,I,1)=PHI(3,I,1)+PHI(1,I,J)*G(J)
  165 IF(K .EQ. 2) PHI(2,I,1)=PHI(2,I,1)+E2AH(I,J)*G(J)
  164 PHI(3,I,1)=PHI(3,I,1)+PHI(2,I,1)
  163 CONTINUE
      DO 166 I=1,N
      DO 166 K=1,N
  166 YN(I)=YN(I)-H*AH2(I,K)*PHI(3,K,1)-ALPHA(1)*EAH(I,K)*Y(1,K)
      GO TO 172
  167 DO 168 I=1,N
      DO 168 K=1,N
  168 YN(I)=YN(I)-ALPHA(1)*EAH(I,K)*Y(1,K)
C***********************************************************************
C*   IF A IS A FUNCTION OF T, PERFORM PERIODIC DECOMPOSITION,          *
C*   EVALUATE A(T(I)), AND ADJUST FINAL Y(N+1)                         *
C***********************************************************************
  172 IF(IAT .EQ. 0) RETURN
  176 TH=T(1)+H
      CALL AFNT(AT,N,TH,NR,NC)
      DO 173 I=1,N
      G(I)=0.0
      DO 173 J=1,N
  173 G(I)=G(I)+(AT(I,J)-A(I,J))*Y(2,J)
      DO 174 I=1,N
      QHI(4,4,I)=0.0
      DO 174 J=1,N
  174 QHI(4,4,I)=QHI(4,4,I)+E2AH(I,J)*G(J)
      DO 175 I=1,N
      DO 175 J=1,N
  175 YN(I)=YN(I)-H*AH2(I,J)*QHI(4,4,J)
      RETURN
  200 CONTINUE
C***********************************************************************
C*   NONLINEAR MULTI-2-STEP STARTS HERE. BEGINNING SECTION DOES        *
C*   INITIALIZATION                                                    *
C***********************************************************************
      DO 240 I=1,N
      DO 241 J=1,N
  241 P1(I,J)=0.0
      YN(I)=0.0
  240 PHI(4,1,I)=0.0
      IF(IS .GT. 1) GO TO 212
C***********************************************************************
C*   EAH CONTAINS EXP(AH), E2AH CONTAINS EXP(2AH)                      *
C***********************************************************************
      IF(N-1) 208,,208
      EAH(1,1)=DEXP(AH(1,1))
      E2AH(1,1)=EAH(1,1)*EAH(1,1)
      GO TO 212
  208 CALL PADE(A,H,EAH,E2AH,E3AH,G,N,NR,NC,INVERT)
      H2=H+H
      CALL PADE(A,H2,E2AH,E3AH,AH4,G,N,NR,NC,INVERT)
  212 IF(INDEX .GT. 0) GO TO 260
C***********************************************************************
C*   EXPLICIT NLM-2-STEP METHODS                                       *
C*   AH2 CONTAINS (AH)**(-2)                                           *
C***********************************************************************
      IF(IS .GT. 1) GO TO 234
  251 IF(N-1) 201,,201
      AH2(1,1)=1./AH(1,1)**2
      GO TO 206
  201 CALL INVERT(AH,N,P1)
      DO 203 I=1,N
      DO 203 J=1,N
      DO 203 K=1,N
```

```
    . 203 AH2(I,J)=AH2(I,J)+P1(I,K)*P1(K,J)
          IF(IT .EQ. 0) GO TO 235
C*************************************************************************
C*    AT THE FINISH OF LOOP 215 --                                        *
C*    PHI(1,I,J) CONTAINS PHI(1,0), PHI(2,I,J) CONTAINS PHI(2,0)          *
C*    FINAL Y(N+1) ARE CALCULATED IN LOOP 232                             *
C*************************************************************************
      206 DO 213 I=1,N
          DO 213 J=1,N
      213 P1(I,J)=ALPHA(1)*E2AH(I,J)+ALPHA(2)*EAH(I,J)+UNIT(I,J)
          DO 215 I=1,N
          DO 215 J=1,N
          DO 215 K=1,N
          PHI(1,I,J)=PHI(1,I,J)+ALPHA(1)*AH(I,K)*E2AH(K,J)
      215 PHI(2,I,J)=PHI(2,I,J)+ALPHA(2)*AH(I,K)*EAH(K,J)
          DO 218 I=1,N
          DO 219 J=1,N
          PHI(1,I,J)=PHI(1,I,J)-P1(I,J)-AH(I,J)
      219 PHI(2,I,J)=PHI(2,I,J)+P1(I,J)+2.*AH(I,J)
      218 P1(1,I)=0.0
      234 DO 220 K=1,KSTEP
          CALL GFN(G,H,N,Y,K,T,A,NR,NC)
          DO 221 I=1,N
          DO 221 J=1,N
      221 YN(I)=YN(I)+PHI(K,I,J)*G(J)*H
      220 CONTINUE
          DO 230 I=1,N
          DO 230 J=1,N
      230 P1(1,I)=P1(1,I)-AH2(I,J)*YN(J)
      235 DO 232 I=1,N
          IF(IT .EQ. 0) P1(1,I)=0.0
          DO 233 J=1,N
      233 PHI(4,1,I)=PHI(4,1,I)-ALPHA(2)*EAH(I,J)*Y(2,J)-ALPHA(1)*E2AH(I,J)*
         $Y(1,J)
      232 YN(I)=P1(1,I)+PHI(4,1,I)
          IF(IAT .EQ. 0) RETURN
      DO 290 I=1,N
      DO 290 J=1,N
  290 E2AH(I,J)=PHI(2,I,J)
      GO TO 176
C**********************************************************************
C*    IMPLICIT NONLINEAR MULTI-2-STEP METHODS                         *
C*    NEXT BELOW 267, AH3 CONTAINS (AH)**(-3)                         *
C**********************************************************************
  260 CONTINUE
      IF(IS .GT. 1) GO TO 284
      IF(N-1) 262,,262
      AH2(1,1)=AH(1,1)*AH(1,1)
      AH3(1,1)=1./(AH2(1,1)*AH(1,1))
      GO TO 263
  262 DO 264 I=1,N
      DO 264 J=1,N
      DO 264 K=1,N
  264 AH2(I,J)=AH2(I,J)+AH(I,K)*AH(K,J)
      DO 267 I=1,N
      DO 267 J=1,N
      AH4(I,J)=0.0
      DO 267 K=1,N
  267 AH4(I,J)=AH4(I,J)+AH2(I,K)*AH(K,J)
      CALL INVERT(AH4,N,AH3)
  263 CONTINUE
  284 IF(IS .GT. 1) GO TO 279
      IF(IT .EQ. 0) GO TO 285
C**********************************************************************
C*    END OF LOOP 275, PHI(L,I,J) CONTAIN PHI(2,L), L=0,1,2           *
C*    FINAL Y(N+1) ARE CALCULATED IN LOOP 282 OR 286                  *
C**********************************************************************
      DO 270 I=1,N
      DO 270 J=1,N
      W(1,1,I,J)=UNIT(I,J)-1.5*AH(I,J)+AH2(I,J)
      W(1,2,I,J)=UNIT(I,J)-0.5*AH(I,J)
      W(1,3,I,J)=UNIT(I,J)+0.5*AH(I,J)
      W(2,1,I,J)=-2.*(UNIT(I,J)-AH(I,J))
      W(2,2,I,J)=-2.*UNIT(I,J)+AH2(I,J)
      W(2,3,I,J)=-2.*(UNIT(I,J)+AH(I,J))
      W(3,1,I,J)=W(1,2,I,J)
      W(3,2,I,J)=W(1,3,I,J)
  270 W(3,3,I,J)=UNIT(I,J)+1.5*AH(I,J)+AH2(I,J)
      DO 283 K=1,3
      DO 272 I=1,N
      DO 273 J=1,N
      DO 274 L=1,N
  274 GHI(K,I,J)=QHI(K,I,J)+ALPHA(1)*W(K,1,I,L)*E2AH(L,J)+ALPHA(2)*W(K,2
     $,I,L)*EAH(L,J)
  273 GHI(K,I,J)=GHI(K,I,J)+W(K,3,I,J)
  272 CONTINUE
  283 CONTINUE
      DO 275 L=1,3
      DO 275 I=1,N
      DO 275 J=1,N
      DO 275 K=1,N
  275 PHI(L,I,J)=PHI(L,I,J)-AH3(I,K)*QHI(L,K,J)
  279 DO 280 I=1,N
      DO 280 K=1,3
      CALL GFN(G,H,N,Y,K,T,A,NR,NC)
      DO 282 J=1,N
      YN(I)=YN(I)+PHI(K,I,J)*G(J)*H
  282 IF(K .EQ. 3) YN(I)=YN(I)-ALPHA(1)*E2AH(I,J)*Y(1,J)-ALPHA(2)*EAH(I,
     $J)*Y(2,J)
  280 CONTINUE
      GO TO 293
  285 DO 286 I=1,N
      DO 286 J=1,N
  286 YN(I)=YN(I)-ALPHA(1)*E2AH(I,J)*Y(1,J)-ALPHA(2)*EAH(I,J)*Y(2,J)
  293 IF(IAT .EQ. 0) RETURN
C*********************************************************************
C*   IF A IS A FUNCTION OF T, PERFORM PERIODIC DECOMPOSITION,         *
C*   EVALUATE A(T(I)), AND ADJUST FINAL Y(N+1)                        *
C*********************************************************************
```

```
  297 T1=T(1)+H
      T2=T1+H
      CALL AFNT(P,N,T1,NR,NC)
      CALL AFNT(AT,N,T2,NR,NC)
      DO 294 I=1,N
      G(I)=0.0
      P1(1,I)=0.0
      DO 294 J=1,N
      G(I)=G(I)+(P(I,J)-A(I,J))*Y(2,J)
  294 P1(1,I)=P1(1,I)+(AT(I,J)-A(I,J))*Y(3,J)
      DO 295 I=1,N
      P1(2,I)=0.0
      DO 295 J=1,N
  295 P1(2,I)=P1(2,I)+PHI(2,I,J)*G(J)+PHI(3,I,J)*P1(1,J)
      IF(KSTEP .EQ. 3) GO TO 298
      DO 296 I=1,N
  296 YN(I)=YN(I)+H*P1(2,I)
      RETURN
  298 DO 299 I=1,N
  299 YN(I)=YN(I)-H*P1(2,I)
      RETURN
  300 CONTINUE
C**************************************************************************
C*   NONLINEAR MULTI=3-STEP STARTS HERE. BEGINNING SECTION DOES          *
C*   INITIAL CALCULATIONS                                                 *
C*   BEFORE 303, THE FOLLOWING RESULTS ARE STORED ...                     *
C*   EAH--EXP(AH),E2AH--EXP(2AH),E3AH--EXP(3AH),AH3--(AH)**(-3)          *
C**************************************************************************
      KUP=KSTEP
      IF(INDEX .EQ. 1) KUP=KSTEP+1
      DO 320 I=1,N
  320 YN(I)=0.0
      IF(IS .GT. 1) GO TO 321
      IF(N-1) 302,,302
      EAH(1,1)=DEXP(A(1,1)*H)
      E2AH(1,1)=EAH(1,1)*EAH(1,1)
      E3AH(1,1)=E2AH(1,1)*EAH(1,1)
      AH2(1,1)=AH(1,1)*AH(1,1)
      IF(INDEX .EQ. 1) GO TO 350
      AH3(1,1)=1./(AH2(1,1)*AH(1,1))
      GO TO 303
  302 DO 330 I=1,N
      DO 330 J=1,N
      DO 330 K=1,N
  330 AH2(I,J)=AH2(I,J)+AH(I,K)*AH(K,J)
      DO 333 I=1,N
      DO 333 J=1,N
      AH4(I,J)=0.0
      DO 333 K=1,N
  333 AH4(I,J)=AH4(I,J)+AH2(I,K)*AH(K,J)
      IF(INDEX .EQ. 1) GO TO 351
      CALL INVERT(AH4,N,AH3)
      CALL PADE(A,H,EAH,E2AH,E3AH,G,N,NR,NC,INVERT)
      H2=H+H
      CALL PADE(A,H2,E2AH,E3AH,AH4,G,N,NR,NC,INVERT)
      H3=H2+H
      CALL PADE(A,H3,E3AH,AH4,W(1,1,1,1),G,N,NR,NC,INVERT)
  303 IF(IT .EQ. 0) GO TO 370
C*********************************************************************
C*   CALCULATE PHI(3,K), K=0,1,2. RESULTS IN PHI(K,I,J)             *
C*********************************************************************
      DO 304 I=1,N
      DO 304 J=1,N
      W(1,1,I,J)=UNIT(I,J)-1.5*AH(I,J)+AH2(I,J)
      W(1,2,I,J)=UNIT(I,J)-0.5*AH(I,J)
      W(1,3,I,J)=UNIT(I,J)+0.5*AH(I,J)
      W(1,4,I,J)=UNIT(I,J)+1.5*AH(I,J)+AH2(I,J)
      W(2,1,I,J)=-2.*(UNIT(I,J)-AH(I,J))
      W(2,2,I,J)=-2.*UNIT(I,J)+AH2(I,J)
      W(2,3,I,J)=-2.*(UNIT(I,J)+AH(I,J))
      W(2,4,I,J)=-2.*UNIT(I,J)-4.*AH(I,J)-3.*AH2(I,J)
      W(3,1,I,J)=UNIT(I,J)-0.5*AH(I,J)
      W(3,2,I,J)=UNIT(I,J)+0.5*AH(I,J)
      W(3,3,I,J)=W(1,4,I,J)
  304 W(3,4,I,J)=UNIT(I,J)+2.5*AH(I,J)+3.*AH2(I,J)
  360 DO 306 II=1,KUP
      DO 307 I=1,N
      DO 308 J=1,N
      DO 309 K=1,N
  309 QHI(II,I,J)=QHI(II,I,J)+ALPHA(1)*W(II,1,I,K)*E3AH(K,J)+ALPHA(2)*W(
     $II,2,I,K)*E2AH(K,J)+ALPHA(3)*W(II,3,I,K)*EAH(K,J)
  308 QHI(II,I,J)=QHI(II,I,J)+W(II,4,I,J)
  307 CONTINUE
  306 CONTINUE
      DO 310 K=1,KUP
      DO 310 I=1,N
      DO 310 J=1,N
      DO 310 L=1,N
      IF(INDEX.EQ.0) PHI(K,I,J)=PHI(K,I,J)-AH3(I,L)*QHI(K,L,J)
  310 IF(INDEX.EQ.1) PHI(K,I,J)=PHI(K,I,J)-AH4(I,L)*QHI(K,L,J)
  321 DO 314 K=1,KUP
      CALL GFN(G,H,N,Y,K,T,A,NR,NC)
      DO 315 I=1,N
      DO 316 J=1,N
C*********************************************************************
C*    CALCULATE FINAL Y(N+1)                                         *
C*********************************************************************
      YN(I)=YN(I)+H*PHI(K,I,J)*G(J)
  316 IF(K .EQ. KUP) YN(I)=YN(I)-ALPHA(3)*EAH(I,J)*Y(3,J)-ALPHA(2)*E2AH(
     $I,J)*Y(2,J)-ALPHA(1)*E3AH(I,J)*Y(1,J)
  315 CONTINUE
  314 CONTINUE
      GO TO 340
  370 DO 371 I=1,N
      DO 371 J=1,N
  371 YN(I)=YN(I)-ALPHA(3)*EAH(I,J)*Y(3,J)-ALPHA(2)*E2AH(I,J)*Y(2,J)-ALP
     1HA(1)*E3AH(I,J)*Y(1,J)
  340 IF(IAT .EQ. 0) RETURN
      IF(INDEX .EQ. 0) GO TO 297
C***********************************************************************
C*   TO SEE IS A A FUNCTION OF T                                       *
C***********************************************************************
      T1=T(1)+H
      T2=T1+H
      T3=T2+H
      CALL AFNT(AT,N,T1,NR,NC)
      CALL AFNT(P,N,T2,NR,NC)
      CALL AFNT(P1,N,T3,NR,NC)
      DO 341 I=1,N
      G(I)=0.0
      QHI(4,3,I)=0.0
      QHI(4,4,I)=0.0
      DO 341 J=1,N
      G(I)=G(I)+(AT(I,J)-A(I,J))*Y(2,J)
      QHI(4,3,I)=QHI(4,3,I)+(P(I,J)-A(I,J))*Y(3,J)
  341 QHI(4,4,I)=QHI(4,4,I)+(P1(I,J)-A(I,J))*Y(4,J)
      DO 342 I=1,N
      DO 342 J=1,N
  342 YN(I)=YN(I)+H*(PHI(2,I,J)*G(J)+PHI(3,I,J)*QHI(4,3,J)+PHI(4,I,J)*QH
     $I(4,4,J))
      RETURN
C***********************************************************************
C*   CALCULATE IMPLICIT PHI FUNCTION, PHI(3,J), J=0,1,2,3              *
C***********************************************************************
  350 AH3(1,1)=AH2(1,1)*AH(1,1)
      AH4(1,1)=1./(AH3(1,1)*AH(1,1))
      GO TO 357
  351 DO 354 I=1,N
      DO 354 J=1,N
      EAH(I,J)=0.0
      AH3(I,J)=AH4(I,J)
      DO 354 K=1,N
  354 EAH(I,J)=EAH(I,J)+AH4(I,K)*AH(K,J)
      CALL INVERT(EAH,N,AH4)
      CALL PADE(A,H,EAH,E2AH,E3AH,G,N,NR,NC,INVERT)
      H2=H+H
      CALL PADE(A,H2,E2AH,E3AH,W(1,1,1,1),G,N,NR,NC,INVERT)
      H3=H2+H
      CALL PADE(A,H3,E3AH,W(1,1,1,1),W(1,2,1,1),G,N,NR,NC,INVERT)
  357 IF(IT .EQ. 0) GO TO 370
      DO 358 I=1,N
      DO 358 J=1,N
      W(1,1,I,J)=-UNIT(I,J)+2.*AH(I,J)-11.*AH2(I,J)/6.+AH3(I,J)
      W(1,2,I,J)=-UNIT(I,J)+AH(I,J)-AH2(I,J)/3.
      W(1,3,I,J)=-UNIT(I,J)+AH2(I,J)/6.
      W(1,4,I,J)=-UNIT(I,J)-AH(I,J)-AH2(I,J)/3.
      W(2,1,I,J)=3.*UNIT(I,J)-5.*AH(I,J)+3.*AH2(I,J)
      W(2,2,I,J)=3.*UNIT(I,J)-2.*AH(I,J)-0.5*AH2(I,J)+AH3(I,J)
      W(2,3,I,J)=3.*UNIT(I,J)+AH(I,J)-AH2(I,J)
      W(2,4,I,J)=3.*UNIT(I,J)+4.*AH(I,J)+1.5*AH2(I,J)
      W(3,1,I,J)=-3.*UNIT(I,J)+4.*AH(I,J)-1.5*AH2(I,J)
      W(3,2,I,J)=-3.*UNIT(I,J)+AH(I,J)+AH2(I,J)
      W(3,3,I,J)=-3.*UNIT(I,J)-2.*AH(I,J)+0.5*AH2(I,J)+AH3(I,J)
      W(3,4,I,J)=-3.*UNIT(I,J)-5.*AH(I,J)-3.*AH2(I,J)
      W(4,1,I,J)=-W(1,2,I,J)
      W(4,2,I,J)=-W(1,3,I,J)
      W(4,3,I,J)=-W(1,4,I,J)
  358 W(4,4,I,J)=UNIT(I,J)+2.*AH(I,J)+11.*AH2(I,J)/6.+AH3(I,J)
      GO TO 360
      END

       SUBROUTINE LMS(KSTEP,H,Y,N,YN,INDEX)
C************************************************************************
C*   LINEAR MULTISTEP METHODS(STEP NUMBER.LE.3)                         *
```

```
C*    BETA COEFFICIENTS ARE FORMULATED TO DEPEND UPON
C*       THE CHARACTERISTIC COEFFICIENTS, ALPHA                          *
C*    100 THRU 400 CALCULATES BETA. CALCULATIONS ARE NEEDED ONLY ONCE    *
C*    BETAS OF EXPLICIT METHODS ARE STORED IN B1. OF IMPLICIT, IN B2     *
C*    THIS SUBROUTINE IS CALLED BY START OR DIFEQ                        *
C************************************************************************
      PARAMETER NR=20, NC=20
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON A(NR,NC),ALPHA(4),T(NR)
      DIMENSION Y(4,NC),YN(N),B(4),B1(3),B2(4),FN(NC)
      DATA IBETA/0/
      K=KSTEP
      DO 1 I=1,N
    1 YN(I)=0.0
      IF(IBETA .GT. 0) GO TO 410
      GO TO (100,200,300),KSTEP
  100 B1(1)=1.0
      B2(1)=0.
      B2(2)=B2(1)
      GO TO 400
  200 B1(1)=0.5*ALPHA(2)
      B1(2)=B1(1)+2.
      B2(1)=5.*ALPHA(2)/12.+1./3.
      B2(2)=2.*ALPHA(2)/3.+4./3.
      B2(3)=-ALPHA(2)/12.+1./3.
      GO TO 400
  300 B1(1)=5.*ALPHA(2)/12.+ALPHA(3)/3.+0.75
      B1(2)=2.*ALPHA(2)/3.+4.*ALPHA(3)/3.
      B1(3)=-ALPHA(2)/12.+ALPHA(3)/3.+2.25
      B2(1)=3.*ALPHA(2)/8.+ALPHA(3)/3.+3./8.
      B2(2)=19.*ALPHA(2)/24.+4.*ALPHA(3)/3.+9./8.
      B2(3)=-5.*ALPHA(2)/24.+ALPHA(3)/3.+9./8.
      B2(4)=ALPHA(2)/24.+3./8.
  400 IBETA=1
      IF(KSTEP .EQ. 1) IBETA=0
  410 IF(INDEX .GT. 0) GO TO 500
```

```
      DO 415 I=1,3
  415 B(I)=B1(I)
  420 DO 450 I=1,KSTEP
      CALL FFN(Y,N,FN,I)
      DO 440 J=1,N
      YN(J)=YN(J)+H*B(I)*FN(J)
  440 CONTINUE
  450 CONTINUE
      IF(K .EQ. KSTEP) GO TO 465
      CALL FFN(Y,N,FN,K)
      DO 460 J=1,N
  460 YN(J)=YN(J)+H*B(K)*FN(J)
  465 DO 470 I=1,N
      DO 470 J=1,KSTEP
      YN(I)=YN(I)-ALPHA(J)*Y(J,I)
  470 CONTINUE
      RETURN
  500 K=KSTEP+1
      DO 510 I=1,4
  510 B(I)=B2(I)
      GO TO 420
      END

      SUBROUTINE PADE(A,H,P,B,C,COL,N,NR,NC,INVERT)
C***********************************************************************
C*    CALCULATE MATRIX EXPONENTIAL BY PADE APPROXIMATION               *
C*    CALLED BY NLMS SUBROUTINE                                        *
C*    CALLS FOR SUBROUTINE INVERT                                      *
C***********************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NR,NC),P(NR,NC),B(NR,NC),C(NR,NC),COL(NC)
      EXTERNAL INVERT
      DATA BETA/.306D0/
      HAVE=H
      DO 2 I=1,N
      DO 1 J=1,N
      B(I,J)=0.0
      C(I,J)=0.0
      P(I,J)=0.0
    1 CONTINUE
      COL(I)=0.0
    2 CONTINUE
      DO 17 I=1,N
      DO 16 J=1,N
      COL(I)=COL(I)+DABS(A(J,I))
   16 CONTINUE
   17 CONTINUE
      XNORM=COL(1)
      DO 18 I=1,N
      IF(XNORM .GT. COL(I)) GO TO 18
      XNORM=COL(I)
   18 CONTINUE
C***********************************************************************
C*   COLUMN NORM IS USED TO SEE WHETHER EXP(A) NEEDS REDUCTION         *
C***********************************************************************
      M=0
   30 IF(XNORM*H-DBLE(.1)) 3,20,20
C***********************************************************************
C*   EXP(A)=(I-(.5+BETA)*A+BETA*A**2)**(-1)*(I+(.5-BETA)*A)            *
C***********************************************************************
    3 DO 6 I=1,N
      DO 5 J=1,N
      DO 4 K=1,N
      P(I,J)=P(I,J)+A(I,K)*A(K,J)
    4 CONTINUE
      C(I,J)=(BETA*P(I,J)*H-(.5D0+BETA)*A(I,J))*H
    5 CONTINUE
      C(I,I)=C(I,I)+1.0
    6 CONTINUE
      CALL INVERT(C,N,B)
      DO 9 I=1,N
      DO 10 J=1,N
      C(I,J)=(.5D0-BETA)*A(I,J)*H
      P(I,J)=0.0
   10 CONTINUE
      C(I,I)=C(I,I)+1.0
    9 CONTINUE
      DO 12 I=1,N
      DO 13 J=1,N
      DO 14 K=1,N
      P(I,J)=P(I,J)+B(I,K)*C(K,J)
   14 CONTINUE
   13 CONTINUE
   12 CONTINUE
      IF(M .EQ. 0) GO TO 40
C***********************************************************************
C*    NORM(AH).GT.(.1), EXP(AH)=EXP(A/2**M)**(2**M)                   *
C***********************************************************************
      DO 24 I=1,N
      DO 25 J=1,N
      B(I,J)=0.0
   25 CONTINUE
   24 CONTINUE
      DO 36 K=1,M
      DO 27 I=1,N
      DO 28 J=1,N
      DO 29 L=1,N
      B(I,J)=B(I,J)+P(I,L)*P(L,J)
   29 CONTINUE
   28 CONTINUE
   27 CONTINUE
      DO 31 I=1,N
      DO 32 J=1,N
      P(I,J)=B(I,J)
      B(I,J)=0.0
   32 CONTINUE
   31 CONTINUE
   36 CONTINUE
```

```
      H=HAVE
      RETURN
   20 H=H/2.0
      M=M+1
      DO 54 I=1,N
      DO 55 J=1,N
      P(I,J)=0.0
   55 CONTINUE
   54 CONTINUE
      GO TO 30
   40 H=HAVE
      RETURN
      END

      SUBROUTINE INVERT(A,N,ANS)
C*******************************************************************
C*   MATRIX INVERSION SUBROUTINE, CALLED BY PADE OR NLMS           *
C*   A CONTAINS THE ORIGINAL ELEMENTS AND REMAINS UNALTERED        *
C*   ANS CONTAINS THE A**(-1)                                      *
C*   THIS SETUP IS USING UNIVAC 1108 MATHPK EXISTING DOUBLE        *
C*      PRECISION GAUSS-JORDAN REDUCTION                           *
C*   THIS PROGRAM IS REPLACEABLE BY THE USER                       *
C*******************************************************************
      PARAMETER NR=20, NC=20
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NR,NC),ANS(NR,NC)
      DIMENSION V(2),JC(NC)
      V(1)=1.D0
      DO 1 I=1,N
      DO 1 J=1,N
    1 ANS(I,J)=A(I,J)
      CALL DGJR(ANS,NR,NC,N,N,$10,JC,V)
      RETURN
   10 WRITE (4,2)
    2 FORMAT(5X,'MATRIX INVERSION ERROR')
      RETURN
      END
      SUBROUTINE GFN(G,H,N,Y,J,T,A,NR,NC)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(NR,NC),Y(4,NC),G(NC),T(NC)
C**********************************************************************
C*   CALCULATES THE G(T,Y). ALL CALCULATIONS TO BE INSERTED HERE     *
C*   CALLED BY NLMS OR DIFEQ                                          *
C*   E.G.     DY/DT=-100Y+(1+T**2)                                    *
C*   DEFINE   G(1)=1.+T(J)*T(J)                                       *
C**********************************************************************
      RETURN
      END


      SUBROUTINE FFN(Y,N,FN,I)
      PARAMETER NR=20, NC=20
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON A(NR,NC),ALPHA(4),T(NC)
      DIMENSION FN(1),Y(4,NC)
C**********************************************************************
C*   CALCULATES THE F(T,Y). ALL CALCULATIONS ENTER HERE               *
C*   CALLED BY SUBROUTINE LMS                                         *
C*   E.G.     DY/DT=-100Y+(1+T**2)                                    *
C*   DEFINE   FN(1)=-100.*Y(1,1)+(1.+T(I)*T(I))                       *
C**********************************************************************
      RETURN
      END

      SUBROUTINE AFNT(A,N,T,NR,NC)
      DOUBLE PRECISION A(NR,NC),T
C**********************************************************************
C*   MATRIX A IS A FUNCTION OF T, ENTER ALL COMPUTATIONS HERE         *
C*   E.G.          A(T)=-100T                                         *
C*   DEFINE        A(1,1)=-100.*T                                     *
C**********************************************************************
      RETURN
      END
      SUBROUTINE PRINT(H,T,Y,N,I)
      DOUBLE PRECISION H,T,Y(1)
C***********************************************************************
C*   DESIGNED FOR USER TO PRINTOUT INTERMEDIATE RESULTS, Y(T)          *
C*   USER CAN DEFINE HIS OWN PRINTOUT FORMATS                          *
C***********************************************************************
   12 FORMAT(1X,2(E15.6,2X),4X,5(E15.8,2X))
 1184 WRITE (4,12) H,T,(Y(J),J=1,N)
   34 CONTINUE
      RETURN
      END


      SUBROUTINE BEGIN(KSTEP,H,N,Y,YN,YOLD,INVERT,IT,METHOD,IAT)
      PARAMETER NR=20, NC=20
      IMPLICIT REAL*8 (A-H,O-Z)
      EXTERNAL INVERT
      COMMON A(NR,NC),ALPHA(4),T(NC)
      DIMENSION Y(4,NC),YN(N),YOLD(4,NC)
C***********************************************************************
C*   DESIGNED FOR USER TO SUPPLY HIS OWN STARTER                       *
C*   ALL CALCULATIONS ENTER HERE                                       *
C***********************************************************************
      RETURN
      END
```

```
C      ***************************************************************
C      *   THIS PROGRAM COMMANDS A SET OF SUBROUTINES TO INTEGRATE A   *
C      *   SYSTEM OF FIRST-ORDER ORDINARY DIFFERENTIAL EQUATIONS OF    *
C      *   THE FORM                                                    *
C      *                       DY(T)/DT=AY + G(T,Y)                    *
C      *                 OR    DY(T)/DT=F(T,Y)                         *
C      *                                                               *
C      *   OVER THE CLOSED INTERVAL-(T(1),TMAX). THIS IS A             *
C      *   FIXED-ORDERING, VARIABLE(FIXED AS WELL)-STEP-SIZE PROGRAM.  *
C      *   ALL INPUTS MUST BE GIVEN. USERS WHO HAVE NO KNOWLEDGE TO    *
C      *   SELECT ALPHA FOR STABILITY, WE SUGGEST ---                  *
C      *     KSTEP=1 -- ALPHA(1),(2)     = -1.D0, 1.D0                 *
C      *     KSTEP=2 -- ALPHA(1),..,(3)  = 0.D0, -1.D0, 1.D0           *
C      *     KSTEP=3 -- ALPHA(1),..,(4)  = 0.D0, 0.D0, -1.D0, 1.D0     *
C      *   ONLY DIFEQ IS CALLED BY THIS PROGRAM, OTHER SUBROUTINES ARE *
C      *   CALLED BY SUBROUTINES. UNUSED SUBROUTINES ARE DEFINED       *
C      *   DUMMIES TO SATISFY THE COMPILER. ALL COMPUTATIONS ARE       *
C      *   PERFORMED IN DOUBLE PRECISION. INPUT PARAMETERS HAVE THE    *
C      *   FOLLOWING DEFINITIONS                                       *
C      *                                                               *
C      *   A       A 2-DIMENSIONAL ARRAY OF (N X N) FOR STORING        *
C      *           ELEMENTS OF MATRIX A                                *
C      *                                                               *
C      *   ALPHA   CHARACTERISTIC POLYNOMIAL COEFFICIENTS. NO CONCERN  *
C      *           TO THE USER. ADAMS COEFFICIENTS ARE SUGGESTED       *
C      *           FOR STRONG STABILITY                                *
C      *                                                               *
C      *   ERR     USER-REQUIRED TOLERANCE                             *
C      *                                                               *
C      *   H       USER-SUGGESTED INITIAL STEP SIZE.                   *
C      *                                                               *
C      *   HMAX    THE MAXIMUM STEP SIZE THE USER ALLOWS THE PROGRAM   *
C      *           TO CONSIDER                                         *
C      *                                                               *
```

```
C     *   IAT     SET =0, IF A IS A CONSTANT                                *
C     *           SET =1, IF A IS A FUNCTION OF T                           *
C     *                                                                     *
C     *   IGFN    SET =0, IF G(T,Y) IS ZERO                                 *
C     *           SET =1, IF G(T,Y) IS NONZERO                              *
C     *                                                                     *
C     *   INDEX   SET =0 FOR EXPLICIT                                       *
C     *           SET =1 FOR IMPLICIT                                       *
C     *                                                                     *
C     *   IPC     SET =0 FOR PREDICTOR OR CORRECTOR                         *
C     *           SET =1 FOR PREDICTOR-AND-CORRECTOR                        *
C     *                                                                     *
C     *                                                                     *
C     *   KSTEP   STEP NUMBER (LESS THAN 4)                                 *
C     *                                                                     *
C     *   METHOD  SET =0, LMS  METHODS                                      *
C     *           SET =1, NLMS METHODS                                      *
C     *                                                                     *
C     *   N       NUMBER OF EQUATIONS                                       *
C     *                                                                     *
C     *   T       A 1-DIMENSIONAL ARRAY, T(1) CONTAINS INITIAL TIME         *
C     *                                                                     *
C     *   TMAX    FINAL TIME, ASSIGNED BY THE USER                          *
C     *                                                                     *
C     *   YZERO   A 1-DIMENSIONAL ARRAY,CONTAINING INITIAL Y                *
C     *                                                                     *
C     *   READ FOLLOWING INPUTS ACCORDING TO USERS PREFERABLE               *
C     *   FORMATS AND HIS ORDER                                             *
C     *     N, KSTEP, IAT, IGFN, INDEX, IPC, METHOD                         *
C     *     H, HMAX, T(1), TMAX, ERR                                        *
C     *     (ALPHA(L+1),L=0,KSTEP)                                          *
C     *     (YZERO(K), K=1,N)                                               *
C     *     ((A(I,J), I=1,N), J=1,N)                                        *
C     ***********************************************************************
      COMMON A(20,20),ALPHA(4),T(20)
      DIMENSION Y(4,20), YZERO(20)
```

```
          DOUBLE PRECISION A,ALPHA,ERR,H,HMAX,T,TMAX,Y,YZERO
          DATA ERR/.1D-11/,H/.1D-2/,HMAX/.1D0/,INDEX/0/,KSTEP/1/,METHOD/1/
C         *****************************************************************
C         *   USER DEFINES HIS INPUTS HERE                                *
C         *****************************************************************
C         10 READ INPUTS
          CALL DIFEQ(ERR,H,HMAX,KSTEP,N,TMAX,Y,YZERO,IGFN,IPC,METHOD,
         *         INDEX,IAT)
C           GO TO 10
          END

          SUBROUTINE START(KSTEP,H,N,Y,YN,YOLD,IT,METHOD,IAT)
C         *****************************************************************
C         *   A SELF-STARTER, CALLED BY DIFEQ                             *
C         *   ARGUMENTS ALREADY DEFINED IN MAIN                           *
C         *   FINAL VALUES ARE STORED IN Y(J,I)                           *
C         *   THIS PROGRAM CALLS 2 SUBROUTINES                            *
C         *           LMS(1,...,0)                                        *
C         *           NLMS(1,...,IAT)                                     *
C         *****************************************************************
          COMMON A(20,20),ALPHA(4),T(20)
          DIMENSION Y(4,20),YN(20),YOLD(4,20)
          DOUBLE PRECISION A,AL,ALPHA,H,HA,T,Y,YN,YOLD
          DATA I0,I1/0,1/
          AL=ALPHA(1)
          ALPHA(1)=-1.D0
          DO 1 I=1,N
        1 YOLD(1,I)=Y(1,I)
          DO 10 J=1,KSTEP
          IF(METHOD .EQ. 1) GO TO 4
C         *****************************************************************
C         *   FIRST-ORDER ADAMS-BASHFORTH METHOD TO START                 *
C         *****************************************************************
          HA=H/4.D0
          DO 3 I=1,4
          CALL LMS(I1,HA,YOLD,N,YN,I0)
```

```
      DO 2 L=1,N
    2 YOLD(1,L)=YN(L)
    3 CONTINUE
      GO TO 6
C     ************************************************************************
C     *   FIRST-ORDER GENERALIZED-ADAMS-BASHFORTH METHOD TO START           *
C     ************************************************************************
    4 CALL NLMS(I1,H,YOLD,N,YN,IQ,I1,IT,IAT)
    6 DO 5 I=1,N
      Y(J+1,I)=YN(I)
    5 YOLD(1,I)=YN(I)
   10 CONTINUE
      ALPHA(1)=AL
      RETURN
      END


      SUBROUTINE DIFEQ(ERR,H,HMAX,KSTEP,N,TMAX,Y,YZERO,IG,IV,METHOD,
     *                 INDEX,IAT)
C     ************************************************************************
C     *   DIFEQ IS CALLED BY MAIN PROGRAM WHOSE ARGUMENTS ARE ALREADY      *
C     *   DEFINED IN THE MAIN PROGRAM WHERE IG=IGFN, IV=IPC                *
C     *                                                                    *
C     *   DIFEQ CALLS 4 SUBROUTINES                                        *
C     *          START(KSTEP,...,IAT) --->A STARTER                         *
C     *          LMS  (KSTEP,...,I)   --->LINEAR MULTISTEP                  *
C     *          NLMS (KSTEP,...,IAT) --->NONLINEAR MULTISTEP               *
C     *          PRINT(TEA,YNEW)      --->FOR USER TO PRINTOUT RESULTS      *
C     *                                                                    *
C     *   SOLUTION VECTOR Y(T) IS YNEW(I) OR Y(KSTEP+1,I)                  *
C     *                                                                    *
C     *   BASED ON USER-SUPPLIED INPUTS, DIFEQ SETS UP THE ITERATIVE       *
C     *   PROCEDURE, CONTROLS STARTER, STEP-SIZE CHANGES, PREDICTOR-       *
C     *   CORRECTOR, CORRECTOR'S CONVERGENCE, PRINTOUTS AND CALLS FOR      *
C     *   THE REQUIRED METHODS                                             *
C     ************************************************************************
```

```
      COMMON A(20,20),ALPHA(4),T(20)
      DIMENSION G(20),Y(4,20),YN(20),YNEW(20),YOLD(4,20),YZERO(20)
      DOUBLE PRECISION A,ALPHA,ANORM,ERR,G,H,HMAX,HMIN,T,TEA,TMAX
      DOUBLE PRECISION TZERO,Y,YN,YNEW,YOLD,YZERO
      DATA HMIN,IZERO,IONE/.1D-11,0,1/
      LMT=3
      WRITE (4,10)
   10 FORMAT(1H1)
      DO 40 I=1,N
   40 Y(1,I)=YZERO(I)
      ITER=0
      ISTEP=0
      TZERO=T(1)
   49 ITER=ITER+1
      IH=0
      IMIN=0
   50 CONTINUE
C     ********************************************************************
C     *   EVALUATE A(T) AT T=T(0) IF IAT NOT 0                           *
C     *   IF KSTEP GT 1, CALLS START                                     *
C     ********************************************************************
      IF(IAT.NE.0) CALL AFNT(A,N,T(1))
      IF(KSTEP.EQ.1 .AND. INDEX.EQ.0) GO TO 60
      CALL START(KSTEP,H,N,Y,YN,YOLD,IG,METHOD,IAT)
   60 TEA=T(1)+DBLE(FLOAT(KSTEP))*H
      IH=IH+1
      DO 61 J=1,KSTEP
   61 T(J+1)=T(J)+H
      IMP=KSTEP+1
      DO 62 J=1,IMP
      DO 62 I=1,N
   62 YOLD(J,I)=Y(J,I)
C     ********************************************************************
C     *    METHOD=0 -- LMS, METHOD=1 -- NLMS                             *
C     *    FIXED-STEP-SIZE     IV=0,    INDEX=0   PREDICTOR              *
C     *                        IV=0,    INDEX=1   CORRECTOR              *
C     *    VARIABLE-STEP-SIZE  IV NE 0,           PREDICTOR-CORRECTOR    *
```

```
C     ******************************************************************
      IF(METHOD.NE.0) GO TO 64
      IF(INDEX.EQ.0) CALL LMS(KSTEP,H,YOLD,N,YN,IZERO)
      IF(IV.EQ.0.AND,INDEX.EQ.1) CALL LMS(KSTEP,H,YOLD,N,YN,IONE)
      GO TO 59
   64 IF(IV.EQ.0.AND,INDEX,EQ.0) CALL NLMS(KSTEP,H,YOLD,N,YN,IZERO,
     *                                     IH,IG,IAT)
      IF(IV.EQ.0.AND,INDEX,NE.0) CALL NLMS(KSTEP,H,YOLD,N,YN,IONE,
     *                                     IH,IG,IAT)
      IF(IV.NE.0) CALL NLMS(KSTEP,H,YOLD,N,YN,IZERO,IONE,IG,IAT)
   59 DO 66 I=1,N
   66 YOLD(KSTEP+1,I)=YN(I)
      IF(IV.NE.0 .OR, IAT.NE.0) GO TO 69
      IF(LMT .EQ. 1) GO TO 69
      DO 68 I=1,N
   68 YNEW(I)=YN(I)
      GO TO 82
C     ******************************************************************
C     *   CORRECTOR AT MOST CORRECT 3 TIMES                            *
C     *   STEP-SIZE CHANGED BY A FACTOR OF 2                           *
C     ******************************************************************
   69 ICORR=0
   70 ICORR=ICORR+1
      IF(ICORR .LE. LMT) GO TO 75
      H=H/2.D0
      IF(H .LT. HMIN) H=HMIN
      IF(H .LT. HMIN) IMIN=IMIN+1
      IF(IMIN .GT. 3) WRITE (4,1170)
 1170 FORMAT(3X,39HH REACHED HMIN, NO CONVERGENCE POSSIBLE)
      IF(IMIN .GT. 3) STOP
      T(1)=TZERO
      DO 71 I=1,N
   71 Y(1,I)=YZERO(I)
      GO TO 50
   75 IF(METHOD.EQ.0) CALL LMS(KSTEP,H,YOLD,N,YNEW,IONE)
      IF(METHOD.GT.0) CALL NLMS(KSTEP,H,YOLD,N,YNEW,IONE,ICORR,IG,IAT)
      IF(LMT .NE. 1) GO TO 76
      DO 77 I=1,N
   77 YNEW(I)=YN(I)
      GO TO 82
   76 CONTINUE
      ANORM=0.D0
C     ***********************************************************************
C     *   TEST CORRECTOR'S CONVERGENCE, USING MAX NORM                      *
C     ***********************************************************************
      DO 80 J=1,N
      G(J)=(YOLD(KSTEP+1,J)-YNEW(J))/YOLD(KSTEP+1,J)
      IF(ANORM .GT. DABS(G(J))) GO TO 80
      ANORM=DABS(G(J))
   80 CONTINUE
      IF(ANORM-ERR) 82,82,1182
 1182 DO 81 J=1,N
   81 YOLD(KSTEP+1,J)=YNEW(J)
      GO TO 70
   82 DO 83 I=1,N
   83 Y(KSTEP+1,I)=YNEW(I)
C     ***********************************************************************
C     *   RESULTS Y(TEA) IN YNEW(I) AND Y(KSTEP+1,I)                        *
C     *   CALL PRINT FOR USER TO PRINT RESULTS                              *
C     ***********************************************************************
      CALL PRINT(H,TEA,YNEW,N,ITER)
   84 CONTINUE
      IF(TEA .GT. TMAX) RETURN
      DO 85 J=1,KSTEP
      DO 85 I=1,N
      Y(J,I)=Y(J+1,I)
      IF(J.EQ.1) YZERO(I)=Y(J,I)
   85 CONTINUE
      T(1)=T(2)
      TZERO=T(1)
      IF(IV.EQ.0 .AND. INDEX.EQ.1) LMT=1
      IF(IV.EQ.0 .AND. INDEX.EQ.1) INDEX=0
      IF(LMT .EQ. 1) IH=0
      IF(IV .EQ. 0) GO TO 60
      T(1)=TEA
      TZERO=T(1)
      DO 86 I=1,N
      YZERO(I)=Y(KSTEP+1,I)
   86 Y(1,I)=Y(KSTEP+1,I)
C     ******************************************************************
C     *   MAINTAIN SUCCESSFUL H CONSTANTLY FOR 3 TIMES BEFORE DOUBLING *
C     ******************************************************************
      ISTEP=ISTEP+1
      IF(ISTEP .LT. 3) GO TO 49
      ISTEP=0
      H=2.D0*H
      IF(H .GT. HMAX) H=HMAX
      GO TO 49
      END

      SUBROUTINE NLMS(KSTEP,H,Y,N,YN,INDEX,IS,IT,IAT)
C     *****************************************************************
C     *   NONLINEAR MULTISTEP ALGORITHM(NLMS)                         *
C     *            CALLED BY DIFEQ OR START                           *
C     *   ARGUMENTS ALREADY DEFINED IN MAIN PROGRAM                   *
C     *   THIS PROGRAM CALLS 3 SUBROUTINES                            *
C     *            INVERT(AH,...,P1)                                  *
C     *            GFN(G,...,NC)                                      *
C     *            PADE(A,...,NC)                                     *
C     *   SOLUTION VECTOR IS STORED IN YN(I)                          *
C     *****************************************************************
      COMMON A(20,20),ALPHA(4),T(20)
      DIMENSION AH(20,20),AH2(20,20),AH3(20,20),AH4(20,20)
      DIMENSION EAH(20,20),E2AH(20,20),E3AH(20,20),G(20)
      DIMENSION P(20,20),P1(20,20),PHI(4,20,20),QHI(4,20,20),UNIT(20,20)
      DIMENSION W1(4,20,20),W2(4,20,20),W3(4,20,20),W4(4,20,20)
      DIMENSION AT(20,20),Y(4,20),YN(20)
      DOUBLE PRECISION A,AH,AH2,AH3,AH4,ALPHA,AT,EAH,E2AH,E3AH,G,H,H2,H3
      DOUBLE PRECISION P,P1,PHI,QHI,T,T1,T2,T3,TH,UNIT,W1,W2,W3,W4,Y,YN
```

```
C      ****************************************************************
C      *   IS IS AN INDICATOR                                         *
C      *   PROGRAM DOES INITIALIZATION WHEN IS IS 1                   *
C      *   PROGRAM CALCULATES AND SAVES AH, EXP(AH), A INVERSE,PHI FN *
C      *   IS.GT.1 ABOVE CALCULATIONS ARE BYPASSED                    *
C      ****************************************************************
       IF(IS .GT. 1) GO TO (100,200,300), KSTEP
       DO 1 I=1,N
            DO 2 J=1,N
            P1(I,J)=0.D0
            UNIT(I,J)=0.D0
            EAH(I,J)=0.D0
            E2AH(I,J)=0.D0
            E3AH(I,J)=0.D0
            AH2(I,J)=0.D0
            AH3(I,J)=0.D0
     2      AH4(I,J)=0.D0
     1 UNIT(I,I)=1.0D0
       DO 3 I=1,4
       DO 3 J=1,N
       DO 3 K=1,N
       PHI(I,J,K)=0.D0
     3 WHI(I,J,K)=0.D0
       DO 6 I=1,N
       DO 6 J=1,N
     6 AH(I,J)=H*A(I,J)
       GO TO (100,200,300) , KSTEP
   100 CONTINUE
C      ****************************************************************
C      *   NONLINEAR MULTISTEP STARTS HERE.                           *
C      *   BEGINNING SECTION DOES INITIALIZATION                      *
C      ****************************************************************
       DO 132 I=1,N
            DO 133 J=1,N
   133      P1(I,J)=0.D0
       YN(I)=0.D0
       PHI(2,I,1)=0.D0
```

```
  132 PHI(3,I,1)=0.D0
      IF(IS.GT.1 .AND. INDEX.EQ.0) GO TO 131
      IF(IS.GT.1 .AND. INDEX.EQ.1) GO TO 170
C     *****************************************************************
C     *   P1 CONTAINS (AH)**(-1), EAH CONTAINS EXP(AH)                *
C     *****************************************************************
      IF(N-1) 120,1120,120
 1120 P1(1,1)=1.0D0/AH(1,1)
      GO TO 122
  120 CALL INVERT(AH,N,P1)
  122 IF(N-1) 123,1123,123
 1123 EAH(1,1)=DEXP(A(1,1)*H)
      GO TO 125
  123 CALL PADE(A,H,EAH,E2AH,E3AH,G,N)
  125 DO 103 I=1,N
      DO 103 J=1,N
  103 AH(I,J)=ALPHA(1)*EAH(I,J)+UNIT(I,J)
C     *****************************************************************
C     *   EXPLICIT NLM-1-STEP METHODS                                 *
C     *   DO LOOP 105 CALCULATES PHI(1,0)                             *
C     *   LOOP 108 OR 110 COMPUTES FINAL Y(N+1)                       *
C     *****************************************************************
      IF(INDEX .NE. 0) GO TO 150
      IF(IT .EQ. 0) GO TO 109
      DO 105 I=1,N
      DO 105 J=1,N
      DO 105 K=1,N
  105 PHI(1,I,J)=PHI(1,I,J)-P1(I,K)*AH(K,J)
  131 CALL GFN(G,H,N,Y,KSTEP,T,A)
      DO 108 I=1,N
      DO 108 J=1,N
  108 YN(I)=YN(I)-ALPHA(1)*EAH(I,J)*Y(1,J)+H*PHI(1,I,J)*G(J)
      RETURN
  109 DO 110 I=1,N
      DO 110 J=1,N
  110 YN(I)=YN(I)-ALPHA(1)*EAH(I,J)*Y(1,J)
      RETURN
```

```
C     *********************************************************************
C     *   IMPLICIT NLM-1-STEP METHODS                                     *
C     *   COMPUTE PHI(1,0), PHI(1,1)                                      *
C     *   FINAL RESULTS ARE CALCULATED IN LOOP 166 OR 168                 *
C     *********************************************************************
  150 IF(N-1) 152,1152,152
 1152 AH2(1,1)=P1(1,1)*P1(1,1)
      GO TO 153
  152 DO 154 I=1,N
      DO 154 J=1,N
      DO 154 K=1,N
  154 AH2(I,J)=AH2(I,J)+P1(I,K)*P1(K,J)
  153 IF(IT .EQ. 0) GO TO 167
      DO 160 I=1,N
            DO 161 J=1,N
                  DO 162 K=1,N
                  PHI(1,I,J)=PHI(1,I,J)+ALPHA(1)*H*A(I,K)*EAH(K,J)
  162       PHI(1,I,J)=PHI(1,I,J)-AH(I,J)
            E2AH(I,J)=AH(I,J)+H*A(I,J)
  161 
  160 CONTINUE
  170 K2=KSTEP+1
      IF(IT .EQ. 0) GO TO 167
      DO 163 I=1,N
            DO 164 K=1,K2
            CALL GFN(G,H,N,Y,K,T,A)
                  DO 165 J=1,N
                  IF(K.EQ.1) PHI(3,I,1)=PHI(3,I,1)+PHI(1,I,J)*G(J)
                  IF(K.EQ.2) PHI(2,I,1)=PHI(2,I,1)+E2AH(I,J)*G(J)
  165 
  164       PHI(3,I,1)=PHI(3,I,1)+PHI(2,I,1)
  163 CONTINUE
      DO 166 I=1,N
      DO 166 K=1,N
  166 YN(I)=YN(I)-H*AH2(I,K)*PHI(3,K,1)-ALPHA(1)*EAH(I,K)*Y(1,K)
      GO TO 172
  167 DO 168 I=1,N
      DO 168 K=1,N
  168 YN(I)=YN(I)-ALPHA(1)*EAH(I,K)*Y(1,K)
```

```
C      *********************************************************************
C      *   IF A IS A FUNCTION OF T, PERFORM PERIODIC DECOMPOSITION,        *
C      *   EVALUATE A(T(I)), AND ADJUST FINAL Y(N+1)                       *
C      *********************************************************************
   172 IF(IAT .EQ. 0) RETURN
   176 TH=T(1)+H
       CALL AFNT(AT,N,TH)
       DO 173 I=1,N
       G(I)=0.D0
       DO 173 J=1,N
   173 G(I)=G(I)+(AT(I,J)-A(I,J))*Y(2,J)
       DO 174 I=1,N
       GHI(4,4,I)=0.D0
       DO 174 J=1,N
   174 GHI(4,4,I)=GHI(4,4,I)+E2AH(I,J)*G(J)
       DO 175 I=1,N
       DO 175 J=1,N
   175 YN(I)=YN(I)-H*AH2(I,J)*GHI(4,4,J)
       RETURN
   200 CONTINUE
C      *********************************************************************
C      *   NONLINEAR MULTI-2-STEP STARTS HERE                              *
C      *   BEGINNING SECTION DOES INITIALIZATION                           *
C      *********************************************************************
       DO 240 I=1,N
            DO 241 J=1,N
   241      P1(I,J)=0.D0
       YN(I)=0.D0
   240 PHI(4,1,I)=0.D0
       IF(IS .GT. 1) GO TO 212
C      *********************************************************************
C      *   EAH CONTAINS EXP(AH), E2AH CONTAINS EXP(2AH)                    *
C      *********************************************************************
       IF(N-1) 208,2208,208
  2208 EAH(1,1)=DEXP(AH(1,1))
       E2AH(1,1)=EAH(1,1)*EAH(1,1)
```

```
      GO TO 212
  208 CALL PADE(A,H,EAH,E2AH,E3AH,G,N)
      H2=H+H
      CALL PADE(A,H2,E2AH,E3AH,AH4,G,N)
  212 IF(INDEX .GT. 0) GO TO 260
C     ******************************************************************
C     *   EXPLICIT NLM-2-STEP METHODS                                  *
C     *   AH2 CONTAINS (AH)**(-2)                                      *
C     ******************************************************************
      IF(IS .GT. 1) GO TO 234
  251 IF(N-1) 201,2201,201
 2201 AH(1,1)=1.0D0/AH(1,1)**2
      GO TO 206
  201 CALL INVERT(AH,N,P1)
      DO 203 I=1,N
      DO 203 J=1,N
      DO 203 K=1,N
  203 AH2(I,J)=AH2(I,J)+P1(I,K)*P1(K,J)
      IF(IT .EQ. 0) GO TO 235
C     ******************************************************************
C     *   AT THE FINISH OF LOOP 215 --                                 *
C     *   PHI(1,I,J) CONTAINS PHI(1,0), PHI(2,I,J) CONTAINS PHI(2,0)   *
C     *   FINAL Y(N+1) ARE CALCULATED IN LOOP 232                      *
C     ******************************************************************
  206 DO 213 I=1,N
      DO 213 J=1,N
  213 P1(I,J)=ALPHA(1)*E2AH(I,J)+ALPHA(2)*EAH(I,J)+UNIT(I,J)
      DO 215 I=1,N
      DO 215 J=1,N
      DO 215 K=1,N
      PHI(1,I,J)=PHI(1,I,J)+ALPHA(1)*AH(I,K)*E2AH(K,J)
  215 PHI(2,I,J)=PHI(2,I,J)+ALPHA(2)*AH(I,K)*EAH(K,J)
      DO 218 I=1,N
           DO 219 J=1,N
           PHI(1,I,J)=PHI(1,I,J)-P1(I,J)-AH(I,J)
  219      PHI(2,I,J)=PHI(2,I,J)+P1(I,J)+2.0D0*AH(I,J)
  218 P1(1,I)=0.D0
  234 DO 220 K=1,KSTEP
      CALL GFN(G,H,N,Y,K,T,A)
      DO 221 I=1,N
      DO 221 J=1,N
  221 YN(I)=YN(I)+PHI(K,I,J)*G(J)*H
  220 CONTINUE
      DO 230 I=1,N
      DO 230 J=1,N
  230 P1(1,I)=P1(1,I)-AH2(I,J)*YN(J)
  235 DO 232 I=1,N
      IF(II .EQ. 0) P1(1,I)=0.D0
      DO 233 J=1,N
  233 PHI(4,1,I)=PHI(4,1,I)-ALPHA(2)*EAH(I,J)*Y(2,J)-ALPHA(1)*E2AH(I,J)*
     $Y(1,J)
  232 YN(I)=P1(1,I)+PHI(4,1,I)
      IF(IAT .EQ. 0) RETURN
      DO 290 I=1,N
      DO 290 J=1,N
  290 E2AH(I,J)=PHI(2,I,J)
      GO TO 176
C     ******************************************************************
C     *   IMPLICIT NONLINEAR MULTI-2-STEP METHODS                      *
C     *   NEXT BELOW 267, AH3 CONTAINS (AH)**(-3)                      *
C     ******************************************************************
  260 CONTINUE
      IF(IS .GT. 1) GO TO 284
      IF(N-1) 262,2262,262
 2262 AH2(1,1)=AH(1,1)*AH(1,1)
      AH3(1,1)=1.0D0/(AH2(1,1)*AH(1,1))
      GO TO 263
  262 DO 264 I=1,N
      DO 264 J=1,N
      DO 264 K=1,N
  264 AH2(I,J)=AH2(I,J)+AH(I,K)*AH(K,J)
      DO 267 I=1,N
```

```
            DO 267 J=1,N
            AH4(I,J)=0.DO
            DO 267 K=1,N
        267 AH4(I,J)=AH4(I,J)+AH2(I,K)*AH(K,J)
            CALL INVERT(AH4,N,AH3)
        263 CONTINUE
        284 IF(IS .GT. 1) GO TO 279
            IF(IT .EQ. 0) GO TO 285
      C     ******************************************************************
      C     *   END OF LOOP 275, PHI(L,I,J) CONTAINS PHI(2,L),L=0,1,2       *
      C     *   FINAL Y(N+1) ARE CALCULATED IN LOOP 282 OR 286              *
      C     ******************************************************************
            DO 270 I=1,N
            DO 270 J=1,N
            W1(1,I,J)=UNIT(I,J)-1.5D0*AH(I,J)+AH2(I,J)
            W1(2,I,J)=UNIT(I,J)-0.5D0*AH(I,J)
            W1(3,I,J)=UNIT(I,J)+0.5D0*AH(I,J)
            W2(1,I,J)=-2.D0*(UNIT(I,J)-AH(I,J))
            W2(2,I,J)=-2.D0*UNIT(I,J)+AH2(I,J)
            W2(3,I,J)=-2.D0*(UNIT(I,J)+AH(I,J))
            W3(1,I,J)=W1(2,I,J)
            W3(2,I,J)=W1(3,I,J)
        270 W3(3,I,J)=UNIT(I,J)+1.5D0*AH(I,J)+AH2(I,J)
            DO 272 I=1,N
                 DO 273 J=1,N
                      DO 274 L=1,N
                      QHI(1,I,J)=QHI(1,I,J)+ALPHA(1)*W1(1,I,L)*E2AH(L,J)
           *                                +ALPHA(2)*W1(2,I,L)*EAH(L,J)
                      QHI(2,I,J)=QHI(2,I,J)+ALPHA(1)*W2(1,I,L)*E2AH(L,J)
           *                                +ALPHA(2)*W2(2,I,L)*EAH(L,J)
                      QHI(3,I,J)=QHI(3,I,J)+ALPHA(1)*W3(1,I,L)*E2AH(L,J)
           *                                +ALPHA(2)*W3(2,I,L)*EAH(L,J)
        274           CONTINUE
                 QHI(1,I,J)=QHI(1,I,J)+W1(3,I,J)
                 QHI(2,I,J)=QHI(2,I,J)+W2(3,I,J)
                 QHI(3,I,J)=QHI(3,I,J)+W3(3,I,J)
```

```
  273       CONTINUE
  272 CONTINUE
      DO 275 L=1,3
      DO 275 I=1,N
      DO 275 J=1,N
      DO 275 K=1,N
  275 PHI(L,I,J)=PHI(L,I,J)-AH3(I,K)*GHI(L,K,J)
  279 DO 280 I=1,N
      DO 280 K=1,3
      CALL GFN(G,H,N,Y,K,T,A)
      DO 282 J=1,N
      YN(I)=YN(I)+PHI(K,I,J)*G(J)*H
  282 IF(K .EQ. 3) YN(I)=YN(I)-ALPHA(1)*E2AH(I,J)*Y(1,J)-ALPHA(2)*EAH(I,
     $J)*Y(2,J)
  280 CONTINUE
      GO TO 293
  285 DO 286 I=1,N
      DO 286 J=1,N
  286 YN(I)=YN(I)-ALPHA(1)*E2AH(I,J)*Y(1,J)-ALPHA(2)*EAH(I,J)*Y(2,J)
  293 IF(IAT .EQ. 0) RETURN
C     ***********************************************************************
C     *   IF A IS A FUNCTION OF T, PERFORM PERIODIC DECOMPOSITION        *
C     *   EVALUATE A(T(I)), AND ADJUST FINAL Y(N+1)                      *
C     ***********************************************************************
  297 T1=T(1)+H
      T2=T1+H
      CALL AFNT(P,N,T1)
      CALL AFNT(AT,N,T2)
      DO 294 I=1,N
      G(I)=0.D0
      P1(1,I)=0.D0
      DO 294 J=1,N
      G(I)=G(I)+(P(I,J)-A(I,J))*Y(2,J)
  294 P1(1,I)=P1(1,I)+(AT(I,J)-A(I,J))*Y(3,J)
      DO 295 I=1,N
      P1(2,I)=0.D0
      DO 295 J=1,N
  295 P1(2,I)=P1(2,I)+PHI(2,I,J)*G(J)+PHI(3,I,J)*P1(1,J)
      IF(KSTEP .EQ. 3) GO TO 298
      DO 296 I=1,N
  296 YN(I)=YN(I)+H*P1(2,I)
      RETURN
  298 DO 299 I=1,N
  299 YN(I)=YN(I)-H*P1(2,I)
      RETURN
  300 CONTINUE
C     ******************************************************************
C     *   NONLINEAR MULTI-3-STEP STARTS HERE                           *
C     *   BEGINNING SECTION DOES INITIALIZATION                        *
C     *   BEFORE 303, THE FOLLOWING RESULTS ARE STORED...              *
C     *   EAH--EXP(AH),E2AH--EXP(2AH),E3AH--EXP(3AH),AH3--(AH)**(-3)   *
C     ******************************************************************
      KUP=KSTEP
      IF(INDEX .EQ. 1) KUP=KSTEP+1
      DO 320 I=1,N
  320 YN(I)=0.D0
      IF(IS .GT. 1) GO TO 321
      IF(N-1) 302,3302,302
 3302 EAH(1,1)=DEXP(A(1,1)*H)
      E2AH(1,1)=EAH(1,1)*EAH(1,1)
      E3AH(1,1)=E2AH(1,1)*EAH(1,1)
      AH2(1,1)=AH(1,1)*AH(1,1)
      IF(INDEX .EQ. 1) GO TO 350
      AH3(1,1)=1.0D0/(AH2(1,1)*AH(1,1))
      GO TO 303
  302 DO 330 I=1,N
      DO 330 J=1,N
      DO 330 K=1,N
  330 AH2(I,J)=AH2(I,J)+AH(I,K)*AH(K,J)
      DO 333 I=1,N
      DO 333 J=1,N
      AH4(I,J)=0.D0
      DO 333 K=1,N
  333 AH4(I,J)=AH4(I,J)+AH2(I,K)*AH(K,J)
      IF(INDEX .EQ. 1) GO TO 351
      CALL INVERT(AH4,N,AH3)
      CALL PADE(A,H,EAH,E2AH,E3AH,G,N)
      H2=H+H
      CALL PADE(A,H2,E2AH,E3AH,AH4,G,N)
      H3=H2+H
      CALL PADE(A,H3,E3AH,AH4,W1(1,1,1),G,N)
  303 IF(IT .EQ. 0) GO TO 370
C     ******************************************************************
C     *   CALCULATE PHI(3,K), K=0,1,2. RESULTS IN PHI(K,I,J)           *
C     ******************************************************************
      DO 304 I=1,N
      DO 304 J=1,N
      W1(1,I,J)=UNIT(I,J)-1.5D0*AH(I,J)+AH2(I,J)
      W1(2,I,J)=UNIT(I,J)-0.5D0*AH(I,J)
      W1(3,I,J)=UNIT(I,J)+0.5D0*AH(I,J)
      W1(4,I,J)=UNIT(I,J)+1.5D0*AH(I,J)+AH2(I,J)
      W2(1,I,J)=-2.D0*(UNIT(I,J)-AH(I,J))
      W2(2,I,J)=-2.D0*UNIT(I,J)+AH2(I,J)
      W2(3,I,J)=-2.D0*(UNIT(I,J)+AH(I,J))
      W2(4,I,J)=-2.D0*UNIT(I,J)-4.D0*AH(I,J)-3.D0*AH2(I,J)
      W3(1,I,J)=UNIT(I,J)-0.5D0*AH(I,J)
      W3(2,I,J)=UNIT(I,J)+0.5D0*AH(I,J)
      W3(3,I,J)=W1(4,I,J)
  304 W3(4,I,J)=UNIT(I,J)+2.5D0*AH(I,J)+3.D0*AH2(I,J)
  360 DO 307 I=1,N
            DO 308 J=1,N
                  DO 309 K=1,N
                  QHI(1,I,J)=QHI(1,I,J)+ALPHA(1)*W1(1,I,K)*E3AH(K,J)
     *                                 +ALPHA(2)*W1(2,I,K)*E2AH(K,J)
     *                                 +ALPHA(3)*W1(3,I,K)*EAH(K,J)
                  QHI(2,I,J)=QHI(2,I,J)+ALPHA(1)*W2(1,I,K)*E3AH(K,J)
     *                                 +ALPHA(2)*W2(2,I,K)*E2AH(K,J)
     *                                 +ALPHA(3)*W2(3,I,K)*EAH(K,J)
                  QHI(3,I,J)=QHI(3,I,J)+ALPHA(1)*W3(1,I,K)*E3AH(K,J)
     *                                 +ALPHA(2)*W3(2,I,K)*E2AH(K,J)
     *                                 +ALPHA(3)*W3(3,I,K)*EAH(K,J)
```

```
                  IF(KUP .NE. 4) GO TO 309
                  QHI(4,I,J)=GHI(4,I,J)+ALPHA(1)*W4(1,I,K)*E3AH(K,J)
     *                                  +ALPHA(2)*W4(2,I,K)*E2AH(K,J)
     *                                  +ALPHA(3)*W4(3,I,K)*EAH(K,J)
  309             CONTINUE
               QHI(1,I,J)=QHI(1,I,J)+W1(4,I,J)
               QHI(2,I,J)=GHI(2,I,J)+W2(4,I,J)
               QHI(3,I,J)=GHI(3,I,J)+W3(4,I,J)
               IF(KUP.EQ.4) QHI(4,I,J)=QHI(4,I,J)+W4(4,I,J)
  308          CONTINUE
  307 CONTINUE
      DO 310 K=1,KUP
      DO 310 I=1,N
      DO 310 J=1,N
      DO 310 L=1,N
      IF(INDEX,EQ.0) PHI(K,I,J)=PHI(K,I,J)-AH3(I,L)*QHI(K,L,J)
  310 IF(INDEX,EQ.1) PHI(K,I,J)=PHI(K,I,J)-AH4(I,L)*QHI(K,L,J)
  321 DO 314 K=1,KUP
      CALL GFN(G,H,N,Y,K,T,A)
      DO 315 I=1,N
      DO 316 J=1,N
C     ************************************************************************
C     *   CALCULATE FINAL Y(N+1)                                             *
C     ************************************************************************
      YN(I)=YN(I)+H*PHI(K,I,J)*G(J)
  316 IF(K .EQ. KUP) YN(I)=YN(I)-ALPHA(3)*EAH(I,J)*Y(3,J)-ALPHA(2)*E2AH(
     $I,J)*Y(2,J)-ALPHA(1)*E3AH(I,J)*Y(1,J)
  315 CONTINUE
  314 CONTINUE
      GO TO 340
  370 DO 371 I=1,N
      DO 371 J=1,N
  371 YN(I)=YN(I)-ALPHA(3)*EAH(I,J)*Y(3,J)-ALPHA(2)*E2AH(I,J)*Y(2,J)-ALF
     1HA(1)*E3AH(I,J)*Y(1,J)
  340 IF(IAT .EQ. 0) RETURN
      IF(INDEX .EQ. 0) GO TO 297
```

```
C     *****************************************************************
C     *   TO SEE IS A A FUNCTION OF T
C     ****************************************************************  *
      T1=T(1)+H
      T2=T1+H
      T3=T2+H
      CALL AFNT(AT,N,T1)
      CALL AFNT(P,N,T2)
      CALL AFNT(P1,N,T3)
      DO 341 I=1,N
      G(I)=0.D0
      QHI(4,3,I)=0.D0
      QHI(4,4,I)=0.D0
      DO 341 J=1,N
      G(I)=G(I)+(AT(I,J)-A(I,J))*Y(2,J)
      QHI(4,3,I)=QHI(4,3,I)+(P(I,J)-A(I,J))*Y(3,J)
  341 QHI(4,4,I)=QHI(4,4,I)+(P1(I,J)-A(I,J))*Y(4,J)
      DO 342 I=1,N
      DO 342 J=1,N
  342 YN(I)=YN(I)+H*(PHI(2,I,J)*G(J)+PHI(3,I,J)*QHI(4,3,J)+PHI(4,I,J)*QH
     $I(4,4,J))
      RETURN
C     *****************************************************************
C     *   CALCULATE IMPLICIT PHI FUNCTION, PHI(3,J), J=0,1,2,3
C     ****************************************************************  *
  350 AH3(1,1)=AH2(1,1)*AH(1,1)
      AH4(1,1)=1.0D0/(AH3(1,1)*AH(1,1))
      GO TO 357
  351 DO 354 I=1,N
      DO 354 J=1,N
      EAH(I,J)=0.D0
      AH3(I,J)=AH4(I,J)
      DO 354 K=1,N
  354 EAH(I,J)=EAH(I,J)+AH4(I,K)*AH(K,J)
      CALL INVERT(EAH,N,AH4)
      CALL PADE(A,H,EAH,E2AH,E3AH,G,N)
      H2=H+H
      CALL PADE(A,H2,E2AH,E3AH,W1(1,1,1),G,N)
      H3=H2+H
      CALL PADE(A,H3,E3AH,W1(1,1,1),W2(1,1,1),G,N)
  357 IF(IT .EQ. 0) GO TO 370
      DO 358 I=1,N
      DO 358 J=1,N
      W1(1,I,J)=-UNIT(I,J)+2.D0*AH(I,J)-11.D0*AH2(I,J)/6.D0+AH3(I,J)
      W1(2,I,J)=-UNIT(I,J)+AH(I,J)-AH2(I,J)/3.D0
      W1(3,I,J)=-UNIT(I,J)+AH2(I,J)/6.D0
      W1(4,I,J)=-UNIT(I,J)-AH(I,J)-AH2(I,J)/3.D0
      W2(1,I,J)=3.D0*UNIT(I,J)-5.D0*AH(I,J)+3.D0*AH2(I,J)
      W2(2,I,J)=3.D0*UNIT(I,J)-2.D0*AH(I,J)-0.5D0*AH2(I,J)+AH3(I,J)
      W2(3,I,J)=3.D0*UNIT(I,J)+AH(I,J)-AH2(I,J)
      W2(4,I,J)=3.D0*UNIT(I,J)+4.D0*AH(I,J)+1.5D0*AH2(I,J)
      W3(1,I,J)=-3.D0*UNIT(I,J)+4.D0*AH(I,J)-1.5D0*AH2(I,J)
      W3(2,I,J)=-3.D0*UNIT(I,J)+AH(I,J)+AH2(I,J)
      W3(3,I,J)=-3.D0*UNIT(I,J)-2.D0*AH(I,J)+0.5D0*AH2(I,J)+AH3(I,J)
      W3(4,I,J)=-3.D0*UNIT(I,J)-5.D0*AH(I,J)-3.D0*AH2(I,J)
      W4(1,I,J)=-W1(2,I,J)
      W4(2,I,J)=-W1(3,I,J)
      W4(3,I,J)=-W1(4,I,J)
  358 W4(4,I,J)=UNIT(I,J)+2.D0*AH(I,J)+11.D0*AH2(I,J)/6.D0+AH3(I,J)
      GO TO 360
      END

      SUBROUTINE LMS(KSTEP,H,Y,N,YN,INDEX)
C     ********************************************************************
C     *   LINEAR MULTISTEP METHODS(STEP NUMBER .LE. 3)                   *
C     *   BETA COEFFICIENTS ARE FORMULATED TO DEPEND UPON THE            *
C     *        CHARACTERISTIC COEFFICIENTS, ALPHA                        *
C     *   100 THROUGH 400 CALCULATES BETA. CALCULATIONS ARE NEEDED       *
C     *        ONLY ONCE                                                 *
C     *   BETAS OF EXPLICIT METHODS ARE STORED IN B1,                    *
C     *          OF IMPLICIT METHODS, IN B2                              *
C     *   THIS SUBROUTINE IS CALLED BY START OR DIFEQ                    *
C     ********************************************************************
```

```
      COMMON A(20,20),ALPHA(4),T(20)
      DIMENSION Y(4,20),YN(20),B(4),B1(3),B2(4),FN(20)
      DOUBLE PRECISION A,ALPHA,B,B1,B2,FN,H,T,Y,YN
      DATA IBETA/0/
      K=KSTEP
      DO 1 I=1,N
    1 YN(I)=0.D0
      IF(IBETA .GT. 0) GO TO 410
      GO TO (100,200,300),KSTEP
  100 B1(1)=1.D0
      B2(1)=0.5D0
      B2(2)=B2(1)
      GO TO 400
  200 B1(1)=0.5D0*ALPHA(2)
      B1(2)=B1(1)+2.0D0
      B2(1)=5.0D0*ALPHA(2)/12.0D0+1.D0/3.0D0
      B2(2)=2.0D0*ALPHA(2)/3.0D0+4.0D0/3.0D0
      B2(3)=-ALPHA(2)/12.D0+1.0D0/3.0D0
      GO TO 400
  300 B1(1)=5.0D0*ALPHA(2)/12.0D0+ALPHA(3)/3.0D0+0.75D0
      B1(2)=2.0D0*ALPHA(2)/3.0D0+4.0D0*ALPHA(3)/3.0D0
      B1(3)=-ALPHA(2)/12.0D0+ALPHA(3)/3.0D0+2.25D0
      B2(1)=3.0D0*ALPHA(2)/8.0D0+ALPHA(3)/3.0D0+3.0D0/8.0D0
      B2(2)=19.0D0*ALPHA(2)/24.0D0+4.0D0*ALPHA(3)/3.0D0+9.0D0/8.0D0
      B2(3)=-5.0D0*ALPHA(2)/24.0D0+ALPHA(3)/3.0D0+9.0D0/8.0D0
      B2(4)=ALPHA(2)/24.0D0+3.0D0/8.0D0
  400 IBETA=1
      IF(KSTEP .EQ. 1) IBETA=0
  410 IF(INDEX .GT. 0) GO TO 500
      DO 415 I=1,3
  415 B(I)=B1(I)
  420 DO 450 I=1,KSTEP
      CALL FFN(Y,N,FN,1)
      DO 440 J=1,N
      YN(J)=YN(J)+H*B(I)*FN(J)
  440 CONTINUE
  450 CONTINUE
      IF(K .EQ. KSTEP) GO TO 465
      CALL FFN(Y,N,FN,K)
      DO 460 J=1,N
  460 YN(J)=YN(J)+H*B(K)*FN(J)
  465 DO 470 I=1,N
      DO 470 J=1,KSTEP
      YN(I)=YN(I)-ALPHA(J)*Y(J,I)
  470 CONTINUE
      RETURN
  500 K=KSTEP+1
      DO 510 I=1,4
  510 B(I)=B2(I)
      GO TO 420
      END

      SUBROUTINE PADE(A,H,P,B,C,COL,N)
C     ******************************************************************
C     *   CALCULATES MATRIX EXPONENTIAL BY PADE APPROXIMATION          *
C     *   CALLED BY NLMS SUBROUTINE                                    *
C     ******************************************************************
      DOUBLE PRECISION A(20,20),P(20,20),B(20,20),C(20,20),COL(20)
      DOUBLE PRECISION BETA,H,HAVE,XNORM
      DATA BETA/.306D0/
      HAVE=H
      DO 2 I=1,N
           DO 1 J=1,N
           B(I,J)=0.D0
           C(I,J)=0.D0
           P(I,J)=0.D0
    1      CONTINUE
      COL(I)=0.D0
    2 CONTINUE
      DO 17 I=1,N
      DO 16 J=1,N
      COL(I)=COL(I)+DABS(A(J,I))
```

```
   16 CONTINUE
   17 CONTINUE
      XNORM=COL(1)
      DO 16 I=1,N
      IF(XNORM .GT. COL(I)) GO TO 18
      XNORM=COL(I)
   18 CONTINUE
C     *****************************************************************
C     *   COLUMN NORM IS USED TO SEE WHETHER EXP(A) NEEDS REDUCTION     *
C     *****************************************************************
      M=0
   30 IF(XNORM*H - .1D0) 3,20,20
C     *****************************************************************
C     *   EXP(A)=(I-(.5+BETA)*A+BETA*A**2)**(-1)*(I+(.5-BETA)*A)        *
C     *****************************************************************
    3 DO 6 I=1,N
            DO 5 J=1,N
                  DO 4 K=1,N
                  P(I,J)=P(I,J)+A(I,K)*A(K,J)
    4             CONTINUE
            C(I,J)=(BETA*P(I,J)*H-(.5D0+BETA)*A(I,J))*H
    5       CONTINUE
      C(I,I)=C(I,I)+1.0D0
    6 CONTINUE
      CALL INVERT(C,N,B)
      DO 9 I=1,N
            DO 10 J=1,N
            C(I,J)=(.5D0-BETA)*A(I,J)*H
            P(I,J)=0.D0
   10       CONTINUE
      C(I,I)=C(I,I)+1.0D0
    9 CONTINUE
      DO 12 I=1,N
      DO 13 J=1,N
      DO 14 K=1,N
      P(I,J)=P(I,J)+B(I,K)*C(K,J)
   14 CONTINUE
   13 CONTINUE
   12 CONTINUE
      IF(M .EQ. 0) GO TO 40
C     *****************************************************************
C     *   NORM(AH).GT.(.1), EXP(A) =EXP(A/2**M)**(2**M)               *
C     *****************************************************************
      DO 24 I=1,N
           DO 25 J=1,N
           B(I,J)=0.D0
   25      CONTINUE
   24 CONTINUE
      DO 36 K=1,M
      DO 27 I=1,N
      DO 28 J=1,N
      DO 29 L=1,N
      B(I,J)=B(I,J)+P(I,L)*P(L,J)
   29 CONTINUE
   28 CONTINUE
   27 CONTINUE
      DO 31 I=1,N
      DO 32 J=1,N
      P(I,J)=B(I,J)
      B(I,J)=0.D0
   32 CONTINUE
   31 CONTINUE
   36 CONTINUE
      H=HAVE
      RETURN
   20 H=H/2.D0
      M=M+1
      DO 54 I=1,N
      DO 55 J=1,N
      P(I,J)=0.D0
   55 CONTINUE
   54 CONTINUE
      GO TO 30
   40 H=HAVE
      RETURN
      END
      SUBROUTINE GFN(G,H,N,Y,J,T,A)
      DOUBLE PRECISION A(20,20),Y(4,20),G(20),T(20),H
C     ******************************************************************
C     *   CALCULATES THE G(T,Y). ALL CALCULATIONS ENTER HERE           *
C     *   CALLED BY NLMS                                               *
C     *   E.G.     DY/DT=-100Y+(1+T**2)                                *
C     *   DEFINE   G(1)=1.+T(J)*T(J)                                   *
C     ******************************************************************
      RETURN
      END
```

```
      SUBROUTINE FFN(Y,N,FN,I)
      COMMON A(20,20),ALPHA(4),T(20)
      DIMENSION FN(20),Y(4,20)
      DOUBLE PRECISION A,ALPHA,FN,T,Y
C     ******************************************************************
C     *   CALCULATES THE F(T,Y). ALL CALCULATIONS ENTER HERE           *
C     *   CALLED BY SUBROUTINE LMS                                     *
C     *   E.G.     DY/DT=-100Y+(1+T**2)                                *
C     *   DEFINE   FN(1)=-100.*Y(I,1)+(1.+T(I)*T(I))                   *
C     ******************************************************************
      RETURN
      END
```

```
      SUBROUTINE AFNT(A,N,T)
      DOUBLE PRECISION A(20,20),T
C     ******************************************************************
C     *   MATRIX A IS A FUNCTION OF T. ENTER ALL COMPUTATIONS HERE     *
C     *   E.G.     A(T)=-100T                                          *
C     *   DEFINE   A(1,1)=-100.*T                                      *
C     ******************************************************************
      RETURN
      END
      SUBROUTINE PRINT(H,T,Y,N,I)
      DOUBLE PRECISION H,T,Y(20)
C     ***************************************************************
C     *   DESIGNED FOR USER TO PRINTOUT INTERMEDIATE RESULTS, Y(T)     *
C     *   USER CAN DEFINE HIS OWN PRINTOUT FORMATS                    *
C     ***************************************************************
   12 FORMAT(1X,2(E15.8,2X),4X,5(E15.8,2X))
 1184 WRITE (4,12) H,T,(Y(J),J=1,N)
   84 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INVERT(A,N,ANS)
C     ***************************************************************
C     *   MATRIX INVERSION SUBROUTINE, CALLED BY PADE OR NLMS         *
C     *   A CONTAINS THE ORIGINAL ELEMENTS AND REMAINS UNALTERED      *
C     *   ANS CONTAINS THE A**(-1)                                    *
C     *   THIS PROGRAM IS REPLACEABLE BY THE USER                     *
C     ***************************************************************
      RETURN
      END
```

INITIAL DISTRIBUTION LIST

Addressee	No. of Copies
ONR, Code 427, 480, 481, 483, 485	5
CNO, Code OP-23T	1
NRL	1
NAVELECSYSCOMQY, Code 03, 051	2
NAVSEASYSCOMHQ, SEA-09G3	4
NAVAIRDEVCEN	1
NAVWPNSCEN	1
DTNSRDC	1
NAVCOASTSYSLAB	1
CIVENGRLAB	1
NELC	1
NUC	1
NAVSEC, SEC-6034	1
NAVPGSCOL	1
APL/UW, Seattle	1
ARL/Penn State	1
DDC, Alexandria	12
Polytechnic Institute of New York, Professor Stanley Preiser	24
Westinghouse Electric Corp., Mr. J. T. Malone	4
Consolidated Edison Company of New York, Inc., Mr. P. Hsiang	1
Mr. R. Casal, Brooklyn, New York	1
RDP Inc., Mr. C. A. Steele, Jr.,	1
Sperry Systems Management, Mr. James Leong	1
Babcock & Wilcox, Mrs. Anne M. Schwartz	1
University of Illinois, Urbana, Dr. S. W. Joshi	1
Clarion State College, Dr. Stephen Gendler	1